基礎演習テキスト

椙山女学園大学現代マネジメント学部　三木　邦弘

2020 年 1 月 14 日版

1. Linux (コンピュータ管理ソフト)

電子機器に組み込まれる超小型のコンピュータを除いたほとんどのコンピュータで、OS (Operating System) と呼ばれるシステム管理ソフトが動いています。OS はコンピュータの持つ資源 (resource: 例えば CPU、メモリ、ファイル、入出力機器) を管理し、そのコンピュータで動くアプリケーションソフトの要求に従い、管理している資源を提供します。パソコンでは Windows が最もよく使われ、それ以外では Unix 系の OS がよく使われています。Unix 系の中では Linux が一番よく使われており、スマホの Android も Linux の一種です。

Windows については大学に入る前からも使う機会があったと思いますが、Linux に関しては Android のスマホぐらいしか縁がないと思います。そして実際 Android のスマホをさわっても、iPhone とあまり変わらないなぁ、という印象を受けるだけでしょう。見た目で言えば Linux は色々です。Windows、iOS (iPhone の OS)、Android などはバージョンが決まればほぼ同じ見た目や操作方法になりますが、Linux は組み合わせ方によって Windows や iOS とよく似た形にもできるし、全くグラフィカルな機能なしでキー入力でコマンドを入れて使うような形にもできます。そのような自由度があるため、スマホでも使われるし、Web サーバーのような、ひたすら送られてきた URL に対してデータを送り返すサービスをするコンピュータでも使われるのです。

1.1 ユーザー登録

複数の人が利用する OS では通常利用する人を登録するユーザー登録が必要です。そして利用者ごとに様々な設定が行われます。システムの管理者であれば、何でもできる権限が与えられ、一般の利用者はシステム関係の設定変更や他の利用者のファイルに触れることができないように設定されます。この演習では mars[*1] という名前のサーバーをみんなで共有して利用します。よって演習を受講する学生を登録するための Web ページを作成しましたので、それを利用してユーザー登録を行います。ブラウザで「https://mars.mgt.sugiyama-u.ac.jp/AC/」へアクセスすると図 1 のようなページが表示されます。

ここで自分の希望する登録名、自分の学籍番号と氏名を入力します。登録名の衝突はあまり考えられませんので、無理に長い複雑なものにする必要はありません。以下は昔の卒論生が使用した登録名です。

aki ako aoki asaka asako bluecat chizuko fujipon fusae happy hiromi ishii itomaki kaori kikumi kitty18 kozue kyoko lovin machi maki mame masami mayo miho mika misa miwako miyabi miyuki momoko naochan natu oida okada rena rie risa saori sasai sasara shiori taki tomo tomoko toshiko toya yuki yukko yuko yunke

1.2 リモートデスクトップによる接続

実は mars 本体には通常マウスやディスプレイが接続されていません。もし接続されていても、複数の学生が一つのディスプレイを覗き込んで作業をするのは大変です。Unix 系の OS は基本的に複数利用者の同時利用が可能になっていて、通常利用者はネットワーク経由で利用します。

昔はネットワーク経由でサーバーに接続し、キーボードでコマンドを入力し、その結果をまた文字出力で得るという形 (CUI: Character User Interface) のみでした。実は現在でもこのような形で管理されているサーバーは少なくありません。一方パソコンなどでは、アイコンやメニューなどを利用した形 (GUI: Graphical User Interface) でのコンピュータ利用が一般的で、同様な形でサーバーの操作ができれば便利です。その代わり CUI であればネットワークを経由して文字のやり取りで済みましたが、GUI では画面の状態なども常時送

[*1] 正式な名称は mars.mgt.sugiyama-u.ac.jp です。なぜ mars かと言えば既に venus があったからです。

図 1　ユーザー登録の画面

らないとマウスポインターも動きません。

Windows では、WindowsXP 以降[*2]に「リモートデスクトップ接続」というソフトが標準でインストールされています。本来は Windows によるサーバーへの接続のためのものですが、サーバーとのやり取りの部分を真似て、Linux であれば xrdp というソフトをインストールすれば、Windows のパソコンから Linux サーバーに接続して使用することができます。さらにスマホ用の「リモートデスクトップ」ソフトもマイクロソフト社から無料で提供されています。よって使いやすいとは言えませんが、スマホから Linux や Windows サーバーを使用することも可能です。

Windows10 では、「スタート」→「Windows アクセサリ」→「リモートデスクトップ接続」で起動することができます。起動すると図 2 のようなダイアログが表示されるので、コンピュータのところに「mars.mgt.sugiyama-u.ac.jp」を入力し接続ボタンをクリックします。

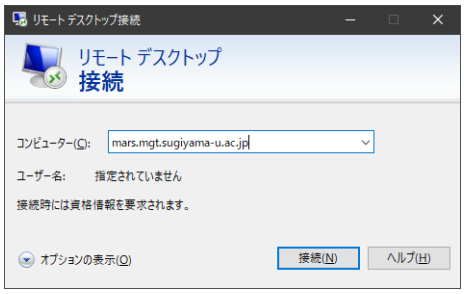

図 2　リモートデスクトップ接続の画面

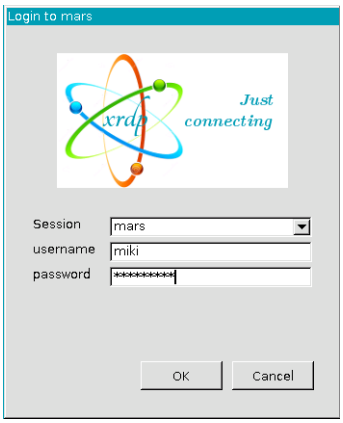

図 3　login の画面

[*2] WindowsXP、Windows vista、Windows7、Windows10 の他、Windows NT 4.0 以降の Windows サーバー

無事に mars に接続できると、図 3 のような mars の認証のダイアログが表示されますので、mars での自分の登録名とパスワードを入力しOKボタンをクリックします。

すると図 4 のような Windows7 以前の Windows に似たデスクトップが表示されます。mars では Linux の中でも ubuntu[*3]と呼ばれるディストリビューションを使っていますが、GUI としては標準の GNOME[*4]ではなく、より軽量で Windows に似ていると言われる LXDE[*5]を利用しています。ただ KDE[*6]や GNOME のソフトも一部取り込んでいるので、本来の LXDE とは少し見た目が異なります。

図 4 LXDE の画面

LXDE の基本的な操作は、

- 画面の左下角にあるスタートボタン？ ( ) をクリックするとメニューが出てくるので、そこで起動したいソフトを選択する。
- 最初のメニューの一番下の「ログアウト」を選択するか、画面右下角にある をクリックすると終了することができる。ログアウトせずにリモートデスクトップ接続を終わると、次回接続した際にすぐに続きをすることができます。
- 漢字を入力する際には、キーボードの半角／全角キーを押すとかな漢字変換が起動される。再度同じキーを押せば元に戻る。

となっています。デスクトップにある「教材フォルダ」と「共有フォルダ」の内容は現マネ棟のパソコン演習室の同名のフォルダと同じ内容で、同じように使用することができます。

[*3] https://ja.wikipedia.org/wiki/Ubuntu 「誰にでも使いやすい最新かつ安定した OS」を開発目標としている。最初から画像編集ソフトやワープロソフト、ゲームなどが入っている。よってデスクトップ OS としてよく使われているが、サーバー用としても使われている。

[*4] https://ja.wikipedia.org/wiki/GNOME Unix 系では X-Window と言う画面表示の基本システムがあり、その上に GNOME、KDE、LXDE などの様々なデスクトップ環境があります。

[*5] https://ja.wikipedia.org/wiki/LXDE

[*6] https://ja.wikipedia.org/wiki/KDE

1.3　パスワードの変更

システムが適当に設定したパスワードでは、入力しにくい、覚えにくいとか、友人に知られてしまったような場合はパスワードを変更しましょう。ウインドウ左下角の「スタート」→「設定」→「パスワード変更」で図 5 のようウインドウが出ますので、現在のパスワードを入力して[OK]をクリックします。すると図 6 のように変わりますので、新しいパスワードを同じものを 2 回入れます。「Password strength meter」のバーが右端まで来るようにするには様々な文字種でかなり長くしなくてはなりませんので、程々でも良いでしょう。また 2 回入力した新パスワードが一致しないと図 6 のように「Password do not match」と出ます。無事入力ができたら再び[OK]をクリックします。

図 5　パスワード変更のウィンドウ

図 6　新パスワード入力のウィンドウ

1.4　インストール済みのソフト

このテキストでは特に取り上げないソフトを簡単に紹介します。「アクセサリ：Galculator」は「アクセサリ」サブメニューにある「Galculator」と言う意味です。

- アクセサリ：Galculator　電卓ソフト
- アクセサリ：Kate 文書ファイル編集ソフト。Windows の「メモ帳」を少し高級にしたもの。
- アクセサリ：nitrogen　壁紙選択ソフト
- アクセサリ：スクリーンショット　画面を画像ファイルに変換する。
- インターネット：firefox ウエッブブラウザ　Web ページ閲覧ソフト
- インターネット：Sylpheed　メールを読み書きするソフト
- オフィス：LibreOffice
  - Base　データベースソフト。Microsoft Access のようなもの。
  - Calc　表計算ソフト。Microsoft Excel のようなもの。
  - Draw　作図ソフト
  - Impress　発表用資料作成ソフト。Microsoft PowerPoint のようなもの。
  - Math　数式記述ソフト
  - Writer　ワープロソフト。Microsoft Word のようなもの。
- グラフィックス：inkspace お絵かきソフト
- システムツール：LXTerminal　Linux のコマンドを入力するためのソフト

2. 文書清書システム

ここでは文書清書システムである LaTeX の使い方を説明します。LaTex は同じく文書清書システムである TeX をより使いやすくしたものですが、まだワープロソフトの生まれる前に誕生したものなので、それでも大変使いにくいものです。イメージとしては、HTML のタグを文章に挿入して Web ページを作成するのに似ています[*7]。HTML のタグは何かして欲しいところに挿入します。LaTeX のコマンドも同様です。よって多少のコマンドは覚えなければなりませんが、ここでは専用の入力ソフト (Texmaker) を使用するので、あまり問題にならないでしょう。

同じコマンドは同じ結果をもたらすために、文書の見た目を揃えることができます。Word で気分で見出しのフォントの大きさを変えたりしていると、簡単に統一の取れていない文書ができてしまいます。このような問題を避けることができます。また HTML のタグと違い、LaTeX のコマンドはカスタマイズが可能です。標準の大きさや形が気に入らない場合は変更することができます。

2.1 Texmaker の起動

LaTex の原稿となるファイルは、Windows 付属の「メモ帖」ソフトでも入力が可能です。HTML のタグと同様に LaTeX のコマンドも普通のキーボードで入力できる文字で構成されているからです。しかし、綴の間違いなどをするとエラーになりますし、入力が終わればコンパイルという処理やその結果の表示などの操作が必要なので、これらを簡単に実行できるソフトが色々作られています。mars では Texmaker というソフトをインストールしていますのでこれを利用します。ウインドウの左下角の「スタート」→「オフィス」→「Texmaker」で起動できます。また拡張子が tex のファイルをダブルクリックしても起動・読み込みができます。

2.2 簡単な例

Texmaker を起動しただけでは文章の入力ができません。「ファイル」→「新規作成」をすると入力できるようになります。簡単な例として以下の文章を入力してください。

```
\documentclass[10pt]{jarticle}
\title{簡単な例}
\author{A18EA999 椙山花子}

\begin{document}
\maketitle

\section{はじめに}

これは大変簡単な例です。文章中の改行は無視されるので、入力の際には
適当に改行してください。1 行空けると段落の終わりとみなされます。

これは 2 つ目の段落です。先頭は勝手に字下げされるので空ける必要はありません。
\end{document}
```

この例を入力する際に \ で始まる横文字の部分 (LaTeX のコマンド) は「LaTeX」メニューで入れることができます。日本語の部分はこの通り入れなくても大丈夫です。無事入力ができたら「ファイル」メニューの「名前をつけて保存」で保存します[*8]。そして F1 のキーを押すとコンパイルの処理が行われて、問題がなけれ

[*7] HTML が Tex の真似をしたという方が正しいが。

[*8] 拡張子として.tex が必要ですが自動的に補われます。

ば PDF 形式の結果ファイルが生成されて、表示されます。

誤りがあった場合には、画面の下に残念ながら英語でメッセージが表示されます。誤りのあった行番号が示されるのでそれを参考に修正をしてください。なお、修正のあとは上書き保存をしなくても、F1のキーを押すとコンパイル処理の前に上書き保存をしてくれます。

2.3 印刷する方法

通常インターネットのどこからでも mars にアクセスすることができます。つまりどこからでも文書作成が可能です。ところが作成した文書を印刷するとなると、mars の側にあるプリンターから出てくるのでは大抵の場合困ります。現在自分のいる場所に近いプリンターから出てくるのが望ましいのですが、それがどれなのかは mars には分かりません。コンパイル処理された文書は PDF 形式のファイルになっているので、これを自分の使っているパソコンにコピーして、パソコンから印刷することによって、身近なプリンターから出すことができます。

1. PDF 形式の文書ファイル (〜.pdf) のアイコンを右クリックする。
2.「コピー」を選択する。
3. パソコンの適当な場所 (例えばデスクトップの空いたところ) で右クリックする。
4.「貼り付け」を選択すると、文書ファイルがパソコンに送られる。卒論のような大きなファイルや画像を多数含むファイルは多少時間がかかるかもしれません。
5. パソコン上の文書ファイルをダブルクリックして開き、そこで印刷の指示をする。

2.4 LaTeX のコマンド

基本的なコマンドの一部を紹介します。なおこの部分は鈴木裕信氏 (hironobu@sra.co.jp) が書かれた「ひろのぶの LaTeX 入門 Version 0.23」を元にかなり省略したものになっています。

LaTeX には局所的なコマンド (command) と広域的な環境 (environment) があります。コマンドは \ で始まります。そして {} や [ ] で囲まれた引数を取ることができます。たとえば、節の見出しのコマンドは \section であり、この章の原稿には、\section{ 文書清書システム } と書いてあります。

一方環境は \begin{*name*} と \end{*name*} で囲まれた範囲が、*name* と言う規則にすべて従うという感じになります。

2.5 使用できる文字

文中で使用できる文字は全角文字、半角の英数字や記号などです。ただし、次の 10 個の特殊文字は、LaTeX の命令に使用されます。ですからそのまま使用することはできません。

$$\# \ \$ \ \% \ \& \ \_ \{ \ \} \ \tilde{} \ \hat{} \ \backslash$$

\ 以外は \#、\$とすることによって文字として表現できます。\ は次のような方法で表現します。

$$\$\backslash\text{backslash}\$ \Longrightarrow \backslash$$

LaTeX はテキスト中に空白があっても自動的につめてしまいます。意図的に空白を挿入する時は、\␣ する必要があります。改行の場合は \\ を使います。マイナス記号 (-) は特別な働きをします。連続して書くことによってハイフンなどの線 (- – —) を引くことができます。

2.6 文章のスタイル

一番最初は必ず \documentclass[*option*]{*style*} で始めなければなりません。そして文章の本体は \begin{document} と、\end{document} で囲まれた範囲内に書きます。

文書のスタイル (*style*) の標準的なものには次のようなものがあります。

	欧文	和文	和文 (新)	和文 (縦書き)
論文	article	jarticle	jsarticle	tarticle
本	book	jbook	jsbook	tbook
報告書	report	jreport	–	treport

「論文」スタイルは最もよく使われます。標準では用紙の下部の中心にページ番号がつきます。このテキストもこのスタイルに手を入れて使っています。\section のレベルから書くことができます。

「本」のスタイルでは、奇数ページの上部 (ヘッダ) では節 (セクション) の番号とセクション名が左側、ページ番号が右側に出ます。偶数ページヘッダでは右側に章 (チャプタ) 名、ページ番号が左側に出ます。章の最初のページが奇数になるようにページが送られます。章立ては必ず \chapter から始めなければなりません。

「報告書」はレポートの形式です。ページ番号は用紙の下部の中心にくるような形になります。章立ては必ず \chapter から始めなければなりません。もし、\section レベルから始めた場合、エラーとなります。

オプション (option) はスタイルの補助的な役割をします。基本となるフォントサイズの変更や見開きに変更する事ができます。

	指定なしの場合	指定できるもの
文字の大きさ	10pt	9pt、10pt、11pt、12pt、14pt、17pt、21pt、25pt、30pt、36pt、43pt
用紙サイズ	A4 サイズ	a4paper、a5paper、b4paper、b5paper
本文の段組	onecolumn (一段組)	onecolumn、twocolumn
見開きの設定	本のスタイルでは twoside、 それ以外では oneside	oneside、twoside
表題を別ページにする	論文スタイルでは本文と同じページに	titlepage

2.7 タイトルページ

タイトルページは \maketitle があらわれた時点で作成されます。よってその前に \title や \author を使って表題や著者を指定する必要があります。\date は指定しなければ、コンパイルを行った日時が使われます。

```
\title{HOW TO USE \LaTeX}
\author{Hironobu Suzuki}
\date{4 Oct 1988}
.....
\maketitle
```

⇒

HOW TO USE LaTeX
Hironobu Suzuki
4 Oct 1988

タイトルページに要約に付ける時は次のようにします。もちろん \maketitle コマンドより前にこれを使う必要があります。

```
\begin{abstract}
This article is for the beginner.
This abstract is printed onto top page.
\end{abstract}
```

⟹

Abstract

This article is for the beginner. This abstract is printed onto top page.

2.8 文章構成

文章の構成で、章や節を表すものとして、次のようなものが用意されています。

\part	\section	\paragraph
\chapter	\subsection	\subparagraph
	\subsubsection	

各々\subsection{**文章構成**}のように見出しの文字を指定します。通常の jarticle スタイルでは、節 (section) の中が項 (subsection) に分かれ、さらにその中が部分項？ (subsubsection) に分かれます。jbook や jreport は章 (chapter) の中が節に分かれ、以下 jarticle と同様に細かく分かれて行きます。

見出しの番号は LaTeX が自動的に付けてくれます。見出しの文字の大きさなども自動的に設定されます。\section を飛ばして\subsection を使用すると番号が変になるのでご注意ください。

2.9 空白

半角の空白は複数あっても、1 つの空白と見なされます。逆に全角の空白は目に見えませんが、行の前後に残っていると歯抜けの段落ができたりするので注意が必要です。

意図的に空白を挿入したい場合は \␣ とします。これは\LaTeX on UNIX のような書き方をしたとき LaTeXon UNIX というような結果になってしまいます。このような結果を避けたい時は\LaTeX\␣on␣UNIX とすれば LaTeX on UNIX となります。

\hspace を使用すると、きっちり指定した幅だけ空けることができます。ただしその空白がちょうど行の先頭や末尾に来たときは無視されますのでご注意ください。

```
1cm \hspace{1cm} space.
1cm \hspace{1cm} space.
1cm \hspace{1cm} space.

This command can accept minus
paramater like this \hspace{-1cm}////
```

⟹

1cm space. 1cm space. 1cm space.

This command can accept minus paramater like/this

2.10 文字のスタイル

文字の大きさは直接ポイント数を与えて変えるような野蛮なことはできません。large や small などのコマンドによって文字のポイント数が変わりますが、原稿の基準フォントのポイントサイズに合わせて相対的に large であり、small であります。このように相対的に指定する事により、後になって基本となるサイズを変更した場合にも、本文の指定を変更する必要がなくなります。

文字の大きさの有効範囲は同一ブロック内です。ブロックとは、環境の中や { と } で囲んだ範囲の事です。例えば、{\Large abc}d と行なうと abc が大きくなり d は元のサイズのままです。実際に行なってみると abcd となります。

通常の文書で頻繁に文字の大きさを変えると見やすさよりも見にくさを助長するだけなので、次の指定はあまり使う機会はないでしょう。

文字 \Huge 文字 \huge 文字 \LARGE
文字 \Large 文字 \large 文字 \normalsize
文字 \small 文字 \footnotesize 文字 \scriptsize
文字 \tiny

英数字の文字のスタイルは色々ありますが、全角に関しては太字と下線のみが有効です。影響を与える範囲などは文字の大きさと同じです。

```
{\bf Bold type style. 全角は太字です。}
{\sf Sans serif type style. 全角は同じです。}
{\sl Slanted type style. 全角は同じです。}
{\tt Typewriter type style. 全角は同じです。}
{\sc Small caps type style. 全角は同じです。}
{\em A emphasized line. 全角は太字です。}
\underline{underlined text} \underline{下線付き文字です。}
```

⟹

Bold type style. 全角は太字です。Sans serif type style. **全角は同じです。***Slanted type style.* 全角は同じです。Typewriter type style. **全角は同じです。**Small caps type style. 全角は同じです。*A emphasized line.* **全角は太字です。**underlined text 下線付き文字です。

下線はコマンドなので指定の仕方が異なりますのでご注意下さい。

2.11 行、段落

原稿の文字列は自動的に詰められ改行されたりします。空行は段落の切れ目を意味しますが、複数の空行は 1 行分の空行と解釈されます。段落の先頭は 1 字下げがなされるので、原稿では字下げは不要です。

文字列や単語の途中で改行が入ってしまった場合は \mbox を使います。例えば、*Segmentation fault* の *Segmentation* と *fault* を離したくない場合は \mbox{*Segmentation fault*} とします。

```
左につめて書きます。ただし、前の行は空白行です。

連続した行です。
連続した行です。
連続した行です。
連続した行です。
連続した行です。
```

⟹

左につめて書きます。ただし、前の行は空白行です。

連続した行です。連続した行です。連続した行です。連続した行です。連続した行です。連続した行です。

2.12　行の空け方

LaTeX では段落、表や画像などがくっつくことがよくあります。これらの間を空けたいときは、\smallskip、\medskip、\bigskip を使用します。実際の間隔はページのレイアウトに左右されますので、同じコマンドでもいつも同じ間隔が空くとは限りません。

行間や段落間の間隔を調整する直接指定する方法として \vspace などのコマンドがありますが、既に書式が決まっているものに無理矢理合わせるような場合以外は使わない方が無難ですので例は省略します。

```
前の行は空白行です。

\smallskip
前の行は空白行です。

\medskip
前の行は空白行です。

\bigskip
前の行は空白行です。
```

⟹

前の行は空白行です。

前の行は空白行です。

前の行は空白行です。

前の行は空白行です。

2.13　中揃え・右揃え

通常のワープロソフトを使用する際でも、中揃えや右揃えは空白を使って位置を調節するのではなく、そういう指示で行うのが常識です。LaTeX では次のようにして中揃えを実現します。

```
\begin{center}
ここにあるものは、中央に寄せられます。

中央
\end{center}
```

⟹

ここにあるものは、中央に寄せられます。
中央

一方 flushright を使用すると右揃えになります。

```
\begin{flushright}
ここにあるものは、右に寄せられます。

右寄り
\end{flushright}
```

⟹

ここにあるものは、右に寄せられます。
右寄り

2.14　箇条書

箇条書の書式として LaTeX では 3 種類の方法があります。先頭に • をつけるもの、先頭に番号を振るものと、先頭にラベルを指定できるものです。それぞれ、itemize、enumerate、description です。

```
\begin{itemize}
\item 各々の行はあたまにポチを付けられてならんでいるでしょ。
\item
書き出しは次の行でも大丈夫です。
\end{itemize}

\begin{enumerate}
\item ここの行の頭には数字がついているはずです。
\item ほらね、2行目でしょ。
\end{enumerate}

\begin{description}
\item [ラベル]　ここの行の頭にはラベルがついているはずです。
\item [Label]　ラベルは bold 体になっています。
\end{description}
```

⟹

- 各々の行はあたまにポチを付けられてならんでいるでしょ。
- 書き出しは次の行でも大丈夫です。

1. ここの行の頭には数字がついているはずです。
2. ほらね、2行目でしょ。

ラベル　ここの行の頭にはラベルがついているはずです。

Label　ラベルは bold 体になっています。

2.15 そのままの形

verbatim はテキストを入力したそのままの形で表すための環境です。例えば、`%\}{`はそのまま打っても LaTeX では正しく表示してくれません。プログラムのリストをそのまま引用するような場合、この環境を使用します。

```
\begin{verbatim}
次のキャラクターは通常、特殊文字として扱われるので表示できない。
バックスラッシュを使うか verbatim を使うかどちらかである。
\ \ \ \ ??? ~~~ $$$$ ###
\end{verbatim}
```

⟹

次のキャラクターは通常、特殊文字として扱われるので表示できない。
バックスラッシュを使うか `verbatim` を使うかどちらかである。
`\ \ \ \ ??? ~~~ $$$$ ###`

行中にこれらの特殊文字を組み込みたいときは、\verb を使用します。\verb+??##$$+ とすると、`??##$$`と出力されます。

2.16 表

LaTeX では、表のようなものを作るのに、tabbing、tabular、array という環境が利用できます。それぞれ想定された用途があり、特徴がありますがここでは tabular のみ説明をします。

tabular の引数の中での命令の意味は、l は左詰め、r は右詰め、c は中央に各項目をそろえます。p{wd} は列の幅を wd に指定 (たとえば p{1cm}) することにより、固定することができます。

| をつけると垂直方向の線が書けます。\\の後の \hline は水平方向の線を引きます。\cline{i – j} は i 列から j 列までの水平方向の線を引きます。

```
\begin{tabular}{||l|c|r||}
\hline
sra & sra-b1-net & 12098 packets \\
      \cline{2-3}
    & sra-2-net  & 290873 packets \\
      \hline
ntt & sun-loop-back  & 2839  packets \\
      \cline{1-1} \cline{3-3}
etl &  & 287 packets \\
\hline
\end{tabular}
```

$\Longrightarrow$

sra	sra-b1-net	12098 packets
	sra-2-net	290873 packets
ntt	sun-loop-back	2839 packets
etl		287 packets

一つ一つの項目欄は \multicolumn{n}{pos}{item} によって調整することができます。n は使用する列数、pos は項目の水平方向の位置、item は項目の文字列です。引数である pos は現在すでに設定されているものを無効とし新たに設定します。この時有効となる引数 pos は l r c | p からなるものでなくてはいけません。

```
\begin{tabular}{|l|l|r|}
\hline \hline
{\em team } &
 \multicolumn{2}{c|}{{\em driver}} \\
\hline
マクラーレン & セナ &プロスト \\
ロータス & ピケ &  中島\\
\hline \hline
\end{tabular}
```

$\Longrightarrow$

team	*driver*	
マクラーレン	セナ	プロスト
ロータス	ピケ	中島

2.17 脚注の入れ方

ページの下の部分に脚注を入れることができます。本文の補足的な情報をここに入れます。例えば URL[*9] などは参考文献として挙げても良いですが、それほど重要でもなければ脚注に入れた方が本文が読みやすくなります。脚注の参照番号が入るところに \footnote{...} を入れます。「...」の部分に脚注の文章が入ります。

2.18 数式

数学モード (math mode) は数式を表現するために使用するモードです。TEX は元もと数学の本を美しく作るために生まれたために数式の表現は大変得意です。しかし、とりあえずは x^2、$\sqrt{2}$ と $\frac{2}{3}$ ぐらいが書ければ十分でしょう。

文中とは独立して数式を示したいときには、`\begin{math}` ... `\end{math}` を使います。文中に数式を入れる場合には`$`...`$`を使います。例えば、`$ \int \sin(2x+3)\cos(x)^{2} dx $`と記述すると、$\int \sin(2x+3)\cos(x)^2 dx$ のように出力されます。

数式中での英字は xy というように斜体になります。一方関数名などは例えば、log は通常の字体でなければなりません。そこで`\log` を使用します。例えば、`$\log x^{y} = y \log x $` とすれば $\log x^y = y \log x$ となります。

[*9] Uniform Resource Locator: 例えば https://ja.wikipedia.org/wiki/Uniform_Resource_Locator

$gcd(x,y)$	`\gcd(x,y)`	$\log(x)$	`\log(x)`
$\max(x,y)$	`\max(x,y)`	$\min(x,y)$	`\min(x,y)`
A^b	`A^{b}`	A_b	`A_{b}`
A_b^c	`A_{b}^{c}`	$\frac{x}{y}$	`\frac{x}{y}`
$\sqrt{A}$	`\sqrt{A}`	$\sqrt[b]{A}$	`\sqrt[b]{A}`

2.19 画像の入れ方

まずいちばん最初の \documsntclass と \begin{document} の間に次のパっケージの設定を入れます。パッケージは LaTeX の機能を拡張するためのもので、これ以外にも様々なもの[*10]があります。

```
\usepackage[dvipdfmx]{graphicx}
```

そして画像を入れたいところに次のようなコマンドを入れます。

```
\includegraphics[width=3cm]{sample-e.png}
```

$\Longrightarrow$ 

この例の場合、「sample-e.png」という画像ファイルの内容が、「width=3cm」によって幅 3cm に整形されて取り込まれます。画像ファイルの形式は、JPEG と PNG のものが可能です。また PDF 形式の文書もこれで取り込むことができます。

画像ファイルは様々な方法で作成することが可能ですが、最終的に原稿のファイルがあるところへ持ってくる必要があります。パソコンまで持ってこれれば「印刷する方法」で PDF ファイルをパソコンに送った方法の逆をすれば、mars に画像ファイルをコピーすることができます。また mars にも画像を編集するソフト「Pinta」[*11]があるので、Windows の「ペイント」のようにお絵かきも可能です。

スマホで撮影した画像を mars に持ってくる場合は、メールに添付して mars へ送る方法があります。またパソコンの画面を画像として取り込む事も可能です。Windows の場合Print Screenボタンを押すと全画面が取り込まれますので、「Pinta」を起動して貼り付けるか、Windows の「ペイント」を起動して貼り付けます。アクティブなウィンドウのみを取り込む場合はAltボタンを押しながらPrint Screenボタンを押します。mars の中のものを画像にする場合はこれらのボタンを利用しても良いですが、「スクリーンショット」[*12]と言うソフトが入っていますのでこちらを利用すると直接画像ファイルになります。

2.20 図表の参照

文書の中に図や表が一つしかない場合はどのような書き方をしても間違いがありませんが、複数の場合はどの図表を指しているのかわかりにくくなります。また図や表のような比較的大きなものの場所を厳密に指定してしまうと、図表がページの残りに入り切らない時に文中に大きな空きができてしまうこともあります。このような問題を生じないように図や表は次のようにしましょう。

図の場合

画像の場合は次のようにします。

[*10] 例えば https://www.biwako.shiga-u.ac.jp/sensei/kumazawa/texindex.html には 300 種類以上のパッケージが紹介されています。

[*11] 画面の左下の「スタート」→「グラッフィクス」→「Pinta」と選択して起動できる。

[*12] 画面の左下の「スタート」→「アクセサリ」→「スクリーンショット」と選択して起動できる。

```
\begin{figure}[htbp]
  \centering
  \includegraphics[width=5cm]{sample-e.png}
  \caption{図の例}
  \label{zu-rei}
\end{figure}
```

最初の行の「htbp」は図をどこに置くかの指定です。「原稿の位置のとおりに表示 (here)」,「ページの上の方に表示 (top)」,「ページの下の方に表示 (bottom)」,「別のページに表示 (page)」の順番に検討するという意味です。\centering は図の位置を中央揃えにする指示です。\caption で図の見出しを指定し、\label でこの図を参照する際の名前を指定します。文中でこの図について触れる箇所に「図`\ref{zu-rei}`」を入れると、「図7」のように表示されます。また「`\pageref{zu-rei}`ページ」を入れると「15 ページ」と言うように図のあるページも表示できます。

図 7　図の例

表の場合

表の場合も tabular 環境を table 環境の中に入れます。figure の部分が table になるだけで、後は図の場合と変わりません。

```
\begin{table}[htbp]
  \centering
  \caption{表の例}
  \begin{tabular}{|l|l|l|}
    \hline
    aaa & bbb & ccc \\
    \hline
    xxx & yyy & zzz \\
    \hline
  \end{tabular}
  \label{hyou-rei}
\end{table}
```

このように caption コマンドを先に使用すると、表の上に見出しを付けることができます。図と表の番号はそれぞれ独立してカウントされます。

表 1　表の例

aaa	bbb	ccc
xxx	yyy	zzz

2.21 参考文献リスト

文書内で、他の本や Web ページなどに掲載された事実や意見などを引用した場合、どこに掲載されていたものか示さないと問題になることがあります。また自分のオリジナルでない意見などは、誰によるものかを明らかにした方が信じてもらえます。

簡易的に出典を示すには既に出てきた脚注を使用しても良いでしょう。もう少しちゃんとしたやり方としては、次のように thebibliography 環境の中で bibitem コマンドで文献を示し、cite コマンドでそれを参照します。

```
参考文献\cite{latex}を参照のこと。

\begin{thebibliography}{99}
  \bibitem{latex の本} Leslie Lamport 著、Edgar Cooke・倉沢良一監訳、「文書処理システム
\LaTeX」、
株式会社アスキー、1990 年
\end{thebibliography}
```

thebibliography 環境で「99」を指定しているのは、参考文献の数が 2 桁という意味です。大抵の場合 99 で十分でしょう。bibitem コマンドで書く内容は、図書の場合、著者名、図書名、出版社、出版年が最低限必要です。これによって次のように示されます。ただし cite コマンドで表示される番号は 2 回コンパイルしないと出てこないので注意が必要です。参考文献 [1] を参照のこと。

参考文献

[1] Leslie Lamport 著、Edgar Cooke・倉沢良一監訳、「文書処理システム LaTeX」 p.191-192、株式会社アスキー、1990 年

3. 三次元モデルの作成

ここでは、三次元モデル作成ソフトである Blender[*13] ver.2.79 の使い方[*14]を説明します。これで作成した三次元モデルは、3D プリンターで実物を作成したり、AR(拡張現実) を利用して現実世界へ投影することができます。三次元モデルを作成するためのソフトは様々なものがあります。Blender は Windows だけでなく Linux などでも使用可能な高機能なフリーソフトです。

Blender は高機能のため様々なことができますが、逆に機能が豊富すぎて困る場合も少なくありません。例えばメニューも多数あり、かつ多段階になっているので順番に開いて探すようなやり方ではまず望みの機能にたどり着けません。また二次元のお絵描きと比べると三次元モデルの作成はかなり複雑と言う点があります。お絵描きソフトで絵を描く場合、二次元のディスプレイの上に見えたものがそのままお絵描きの結果となります。三次元の場合、二次元のディスプレイに見えるものはある方向から見た場合の姿ですので、別の方から見ると全然予定と異なる形になっていることも少なくありません。また色についてもお絵描きでは、色を選択して塗りつぶすという感じですが、三次元では材質 (マテリアル) の色 (ディフューズ) を設定し、さらに鏡面反射 (スペキュラー) する色、反射する割合なども設定します。光線の位置や強さも別に設定することにより、他の物体の映り込みなどは自動的に計算されます。逆にお絵描きでは映り込みなどは自分で描き込まなければいけません。

さらに三次元モデルでは動かす事も可能になっています。アニメの登場人物を三次元モデルで作成する際には、モデルの内部に骨や関節を設定し、体の各部分をシーンに合わせて動かせるようにします。お絵描きでアニメを作成する場合は、少しずつ動いたシーンを描くことになりますが、三次元の場合は一つのモデルの変形で対応できます。さらに MikuMikuDance[*15]のような動画作成ソフトでは、最初と最後の姿勢を指定すると、その間の動きを自動的に補間してくれますが、二次元の絵で同様のものはありません。

3.1 Blender の起動方法

パソコンで Blender を使用する場合は、そのパソコンに Blender を自分でインストールする必要があります。mars には既にインストールしてありますので、すぐに使うことができます。ただし複数の人が同時に使うと処理が追いつかなくなって遅くなる可能性があります。mars ではウインドウの左下角の「スタート」→「グラフィックス」→「Blender」で起動できます。すると図 8 のような Blender の画面が出てきますが、かなり複雑です。中央のオブジェクトの表示画面の周りを様々なメニューやボタンが取り囲んでいます。

Blender で作成する三次元モデルを構成するものをオブジェクトと呼びます。各オブジェクトは平面で構成されています。曲面ではありません。球のような丸いものは、実用上問題のないレベルまで細かく分割した平面にします。

Blender では様々な操作をメニューで選択して行うことができますが、キー入力でもかなりの操作を指示することができます。メニューで一々選択するよりもその方が早いのでキー入力で行う方を勧めますが、マウスポインターが適切な場所にないとうまく行きませんのでご注意ください。

Blender ではよく右クリックを使用します。例えばオブジェクトやオブジェクトを構成する面・線・点を選択するのも右クリックです。また操作を間違えたらすぐ Ctrl+Z で戻しましょう。

[*13] https://www.blender.org/

[*14] 他のソフトでもよくあることだが、バージョンが異なると操作法が変わってしまうので、検索して使い方を探す場合はどのバージョンの説明なのかに注意しましょう。ちなみに既に ver.2.80 が出ていますが色々変わっているようです。

[*15] https://sites.google.com/view/vpvp/

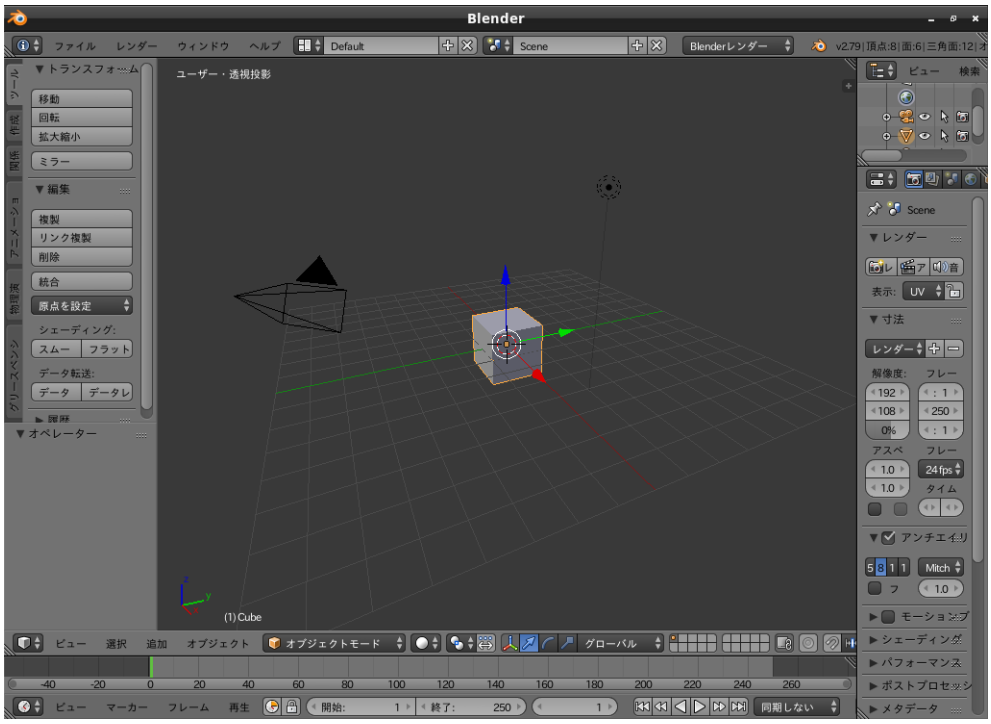

図 8　Blender の画面

3.2　シーンに対する操作

全体をシーンと呼びます。シーンには、複数のオブジェクトだけでなく、照明やカメラが含まれます。マテリアルにいくら色を設定しても発光する設定がなく、照明がないと真っ黒です。同じ色でも照明の当て方によって明るさが変わりますし、影も生じます。よって Blender で綺麗なオブジェクトの画像を作成したい場合は、照明をどのように当てるかは大きな問題となります。一方オブジェクトを取り出して 3D プリンターで造形する場合は、Blender の外で様々な光に当たる訳ですから、照明の設定は不要です。カメラの位置や向きなどの設定もオブジェクトの画像を作成したい場合の際は重要ですが、オブジェクトを外へ取り出す場合は関係ありません。

3.2.1　寸法の単位の設定

オブジェクトを 3D プリンターで実物にする場合や AR(拡張現実) で設定した大きさで表示したい場合、そまず単位を設定する必要があります。右側のパネルのところで図 9 のように設定をします。まず  (プロパティ) を選択し、次に  (シーン) を選択すると「単位」が設定できるようになります。3D プリンターや AR を利用する場合は「単位」の選択を通常「ミリメーター」(メニューでの選択は「millimeters」) にします。

3.2.2　シーンを見る向きの変更

マウスポインターを中央のオブジェクトの表示の部分へ持っていって、テンキーの数字を押すと表 2 のようにオブジェクトの見える向き (視点) が変わります。キーボードのアルファベットの上に横一列に並んでいる数字のキーではだめなのでご注意ください。テンキーを押す際にCtrlキーを押しながらだと回転などの向きが逆になります。Shiftキーを押しながらだと選択対象基準で回転などをするようになります。

マウスのホイールで視点からの距離を変えることができます。近くにすればオブジェクトが大きく見え、遠くにすれば小さく見えます。またCtrlを押しながらマウスのホイールを動かすと視点を左右に、Shiftを押しながらマウスのホイールを動かすと視点を上下に動かすことができます。

図 9　寸法の単位の設定

表 2　テンキーによる視点の変更

キー	視点
2	下方向に 15 ° 回転
8	上方向に 15 ° 回転
4	左方向に 15 ° 回転
6	右方向に 15 ° 回転
9	180 ° 回転
1	正面からにする
3	右正面からにする
7	真上からにする
0	カメラからの視点にする
5	投影方法の切り替え
.	表示されているオブジェクトの全てが見えるようにする
/	選択しているオブジェクトが見えるようにする

3.3　オブジェクトに対する操作

Tabで「オブジェクトモード」と「編集モード」の切り替えができます。「オブジェクトモード」では、次に述べるようなオブジェクトの位置などオブジェクト丸ごとの操作ができます。最初はオブジェクトモードになっているので、中央のオブジェクトの表示画面で右クリックで操作するオブジェクトを選択します。

3.3.1　オブジェクトの追加

左パネルの「作成」タブのところに、「平面」、「立方体」、「円」、「UV 球」などのオブジェクトを作成するボタンがあります。クリックすると 3D カーソルの位置に新しいオブジェクトができます。場合によっては大きすぎたり、小さすぎるオブジェクトが出てきます。その場合は左パネルの下のところにある半径の数字を変更して直します。

3.3.2 文字オブジェクトの追加

立体的な文字を作成することができます。なお、説明文のような文章は後述のテクスチャを利用して、文章の画像を貼り付けた方が良いかもしれません。画面の真ん中でShift+A でメニューが出ます。ここで「フォント」を選択すると、カーソルのあった位置に「Text」と言う厚みのない文字オブジェクトが追加されます。Tabで「編集モード」に変更するとBack Spaceで文字を消したり、キー入力で文字を追加することができます。

文字に漢字などが含まれる場合は、フォントを変更しないと表示されません。右のパネルのところで、  (プロパティ) を選択し、次に  (フォント) を選択します。その下の「フォント」の標準の 

の部分の F と X の間のボタンをクリックするとファイルの指定画面になります。ここで、フォルダーを順番に usr、share、fonts、truetype と開いていくと fonts-japanaese-mincho.ttf や gonts-japanese-gothic.ttf が出てくるのでこのどちらかを選択すると日本語が表示されるようになります。

同じところで、「押出」に 0 より大きな値を設定すると厚みのある立体的な文字になります。文字の大きさや文字間隔を変更する際には、Tabで「オブジェクトモード」に戻ってから下部のパネルにある  の四角から左下に棒が伸びているボタンをクリックして文字オブジェクトから出ている緑や赤の矢印をドラッグすると大きさが、青の矢印をドラッグすると文字の厚みを変更することができます。いつものオブジェクトを移動させる矢印に戻す場合は、2 つ右にある矢印のボタンをクリックします。

位置以外の形状などの設定が終わったら、文字オブジェクトの形式をメッシュに変更します。これをしないと立方体などのオブジェクトを結合などができませんし、3D プリンターなどで造形もできません。メッシュに直すには、Alt+C を押して出てきたメニューで「カーブ/メタ/サーフェス/テキスト からメッシュ」の方を選択します。するとこれまで固まりだった文字オブジェクトが平面で囲われた形になります。一度メッシュにすると元に戻すことはできません。フォントの変更が必要となったらもう一度文字オブジェクトの追加からしましょう。

3.3.3 オブジェクトの名前の設定

新しく追加されたオブジェクトには自動的に名前が付いています。機械的に立方体ならば「Cube001」のような名前が付くので、どんどん変形をして全く違う形になった場合識別する役に立ちません。オブジェクトを右クリックして選択すると、右側のパネルの一番上の「ビュー」のところにある、対応するオブジェクトの名前が白くなるので、そこを右クリックすると出てくるメニューの中に「名前変更」があります。

3.3.4 数字でオブジェクトの大きさや位置を設定

オブジェクトを 3D プリンターで実物にする場合、その大きさの設定をマウスで適当に済ますわけには行きません。数字できっちり設定するためにはまず単位を設定する必要があります。そしてオブジェクトを選択して N を押すと右側に設定用のパネルが出てきます。そこで位置、回転、寸法を数字で指定できます。「拡大・縮小」のところは寸法を変更すると自動的に元の大きさに対する倍率が設定されます。逆にここで倍率を入力すると「寸法」のところの数字が自動的に設定されます。

なおこのパネルが不要になった場合は、パネルの左側の境界を右へドラッグすると消すことができます。

3.3.5 オブジェクトの移動

オブジェクトを選択すると赤、緑、青の矢印が出てきます。これはそのオブジェクトの原点から x 軸、y 軸、z 軸の伸びる方向を示します。青の矢印の向いている方向が z 軸なので上方向になります。この矢印をドラッグするとオブジェクトの位置をその軸の方向へ動かすことができます。これ以外の方法として、オブジェクトを右ドラッグ (マウスの右ボタンを押しながら動かすこと) で移動し、最終的な場所に到達したらクリックして移動を確定させる方法もあります。しかし、三次元の世界を二次元のディスプレイで示しているため、なかな

か思った方向にオブジェクトが動いてくれません。

位置を数字で指定したい場合は、N を押すと右側に出てくる設定用のパネルでできます。

3.3.6　オブジェクトの削除

右クリックでオブジェクトを選択して、X を押すと、削除するかどうか尋ねてきます。「削除」をクリックするとオブジェクトが削除されます。

3.3.7　オブジェクトの複製

右クリックでオブジェクトを選択して、Shift+D でオブジェクトの複製ができます。できたオブジェクトは元のオブジェクトと重なっているので、マウスをそのまま動かすと出てきます。適当な位置まで動かしたらクリックすると位置が確定します。

3.3.8　オブジェクトの結合

Shiftキーを押しながら右クリックすると、複数のオブジェクトを選択することができます。そしてCtrl+Jで選択した複数のオブジェクトを一つのオブジェクトにすることができます。なお、この方法で結合したオブジェクトは、あとで元のオブジェクトに分けることができます。

3.3.9　オブジェクトの計算

オブジェクトに丸い穴を開けたい場合、オブジェクトを後述の変形操作で丸く削るのは困難です。一方丸い穴の形状は通常円筒で、この形のオブジェクトは簡単に作ることができます。よって元のオブジェクトから円筒オブジェクトの部分を引き算すれば、丸い穴の空いたオブジェクトを簡単に作ることができます。このようにあるオブジェクトに別のオブジェクトを加えたり (結合)、引いたり (差分)、重なり部分だけにしたり (交差) することができます。

まず元のオブジェクトを右クリックで選択し、右のパネルのところで、  (プロパティ) を選択し、次に  (モディファイヤ) を選択すると追加が出てくるのでクリックするとメニューが表示されます。その中から「ブーリアン」を選択します。「演算」のところで、「差分」、「結合」、「交差」を選択し、「オブジェクト」の下のところをクリックするともう一つのオブジェクトが選択できます。すると画面には計算結果が表示されるので、それで問題なければ適用をクリックします。

3.4　オブジェクトの変形などの操作

Tabで「オブジェクトモード」と「編集モード」の切り替えができます。「オブジェクトモード」ではオブジェクトの位置などオブジェクト丸ごとの操作ができますが、「編集モード」ではオブジェクトの部分に対する操作が可能です。これを編集モードにすると、画面下部に  が出てきて、ここで左から点、線。面の選択を示しているので、適当なものをクリックしてから、右クリックで、オブジェクトを構成する点、線や面を個別に選択できます。その後オブジェクトの移動と同じ操作を行うと、選択した点、線や面のみが動くので、オブジェクトの形が変わる事になります。

3.4.1　オブジェクトの底面を原点に移動

3D プリンターで出力する 3 次元モデルは宙に浮いていては困ります。オブジェクトの底面となる面を選択し、Shift+S でメニューが出るので「カーソル→選択物」を選択して 3D カーソルを底面の中央に移動し、「オブジェクト・モード」に変更して左パネルの「ツール」タブの中の原点を設定をクリックし、「原点を 3D カーソルへ移動」を選択します。この場合原点がオブジェクトの方にやってくるので、見た目は変わりません。N を押して位置の数字を全てゼロにすると正しく設定できたかどうかわかるでしょう。

3.4.2 面の細分化

元となるオブジェクトが少ない数の面で構成されていると、形を細かく設定することができません。そのような場合は面を細分化するとやりやすくなります。オブジェクトの細分化したい面を右クリックで選択し、W、1(数字のいち) の順に押すと細分化されます。繰り返すとさらに細分化されます。このやり方では長方形に分割されますが、面を選択した上でAlt+P で扇形に分割することもできます。

3.4.3 角を丸める

我々が手にするものは大抵角が丸く作られています。そうしないと手などに当たった時に怪我をする恐れがあるからです。オブジェクトの角を丸くするには、面や線を選択した上で、Ctrl+B を押すと「ベベル」と言う面取りの状態になります。マウスを動かすとどのくらいの幅を丸くするか (量) が変わり、ホイールを動かすと分割数が変わります。適当なところでクリックすると確定します。すると左側の「ベベル」のところに、量、セグメント (分割数)、側面 (どのくらい内側に凹ますか) の数値が表示されますので、ここでキー入力で設定することができます。

3.4.4 マテリアルの追加

Blender を起動したり、初期化した際にある立方体には既にマテリアルが設定されています。マテリアルはオブジェクトの材質であり、オブジェクトの色などはマテリアルに設定します。複数のオブジェクトがある場合、同じマテリアルのオブジェクトが合っても構いません。例えば機械のネジなどは、大きさや形状は異なっても、通常同じ材質材質ですので同じマテリアルを設定します。

3.4.5 テクスチャの貼り付け

オブジェクトの表面に画像を貼り付けることによりオブジェクトの質感 (ザラザラしているとか) や模様を付けることができます。この貼り付けられた画像のことをテクスチャと呼びます。ただ画像は平面ですが、オブジェクトの表面は多角形です。よって通常のオブジェクトにテクスチャを設定するには、そのオブジェクトの展開図を作成し、それに合った画像を用意するような手順となります。画像の作成は Blender で行うこともできますし、一旦別のソフトで編集したものを取り込むことも可能です。なおテクスチャはオブジェクトに直接結びつくのではなく、マテリアルに付く形になるので、もしオブジェクトにマテリアルが設定されていない場合は、まずマテリアルを設定する必要があります。

3.5 オブジェクトなどの保存

三次元モデルが完成したら、上部にある「ファイル」メニューの中の「名前を付けて保存」を選択します。そして保存場所やファイル名を設定して保存します。拡張子は「.blend」が付きます。これはシーン全体の保存となりオブジェクトだけでなく照明やカメラなども含まれます。

3D プリンターで使用する STL 形式で保存する際は、右クリックで保存するオブジェクトを選択してから、上部にある「ファイル」メニューの中の「エクスポート」の中の「Stl (.stl)」を選択します。拡張子は「.stl」が付きます。AR で使用する OBJ 形式も同様にオブジェクトを選択してから、「ファイル」→「エクスポート」→「Wavefront (.obj)」を選択して保存します。拡張子は「.obj」が付きます。

3.6 プレートの作成

個々の操作の説明だけでは、なかなか実際のオブジェクトの作成方法は分かりません。ここでは図 10 のような名前プレートの作成を例にして、Blender でいかに三次元モデルを作成するかを説明します。なお () の間の数字はやり方を説明している章の番号です。

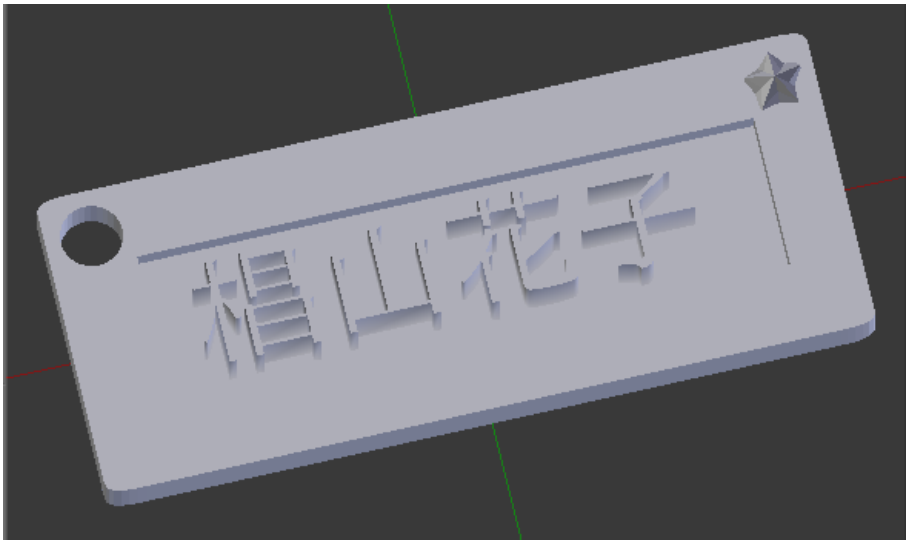

図 10　名前プレート

1. Blender を起動して、単位を mm に設定する。(3.2.1)
2. 最初からある立方体 (Cube) の大きさを 50mm×20mm×3mm にする。(各々 xyz の値)(3.3.4)
3. Cube の下の面の中央を原点にする。(3.4.1)
4. Cube の 4 つの角を丸くする。量は 0.1、セグメントは 8、側面は 0.8 にする。(多少異なる値でも構わないが 4 つの角が全て同じ丸みになるようにする)(3.4.3)
5. 立方体 (Cube001) を追加しその大きさを 40mm×10mm×3mm にする。(3.3.1)
6. Cube001 を動かして Cube に重ねる。Cube001 の位置を x=0.5mm、y=0、z=1.5mm にする。(3.3.5)
7. Cube から Cube001 と重なった部分を切り取る。そして Cube001 をずらしてちゃんと凹みがついていれば、Cube001 を削除する。(3.3.9)
8. 半径 2mm、長さ (深度)10mm 程度の円柱を追加する。これを Cube の左上の部分に重ねた (x=-22mm、y=7mm) 上で、重なった部分を切り取る。うまく Cube に丸穴が空いたら円柱を削除する。(3.3.9)
9. 星形を付ける。半径 2mm、長さ (深度)1mm、頂点数 10 の円柱を追加する。円柱の上面の頂点を一つおきに内側に動かし星形にする。さらに上面を扇形に分割し、中心点を少し持ち上げて立体的な星形にする。これを Cube に埋め込んで結合させる。(3.4.2)
10. 文字オブジェクト (椙山花子) を追加する。押し出しを 1mm にする。また文字の大きさは Cube の凹みに入り、丸穴や星形と重ならない程度にする。メッシュ形式に変換した上で z=0.5mm にして Cube に埋め込み、結合し、文字オブジェクトを削除する。(3.3.2)
11. 名前を付けて保存する。(3.5)
12. stl 形式で保存して 3D プリンターで造形する。(宙に浮いた部分はないがサポートは追加する)(3.5)
13. obj 形式で保存して AR で表示する。(3.5)

4. 3D プリンター

通常のプリンターは平面の紙の上に字や絵を書きます。3D プリンターは立体を作ることができます。様々な方式のプリンターがあり、立体を作成するために使用する物質によってもできるものが変わります。この授業で使用する 3D プリンターは、比較的低い温度で柔らかくなるプラスチックを利用して、紐状の粘土で土器を作るように立体を造形します。色はプラスチックの色になりますので、通常は単色の立体になります。高級な 3D プリンターでは途中で別の色のプラスチックに切り替えることによって別の色の部分を作ることができますが、何種類もの色というわけには行きません。しかし、比較的構造が簡単で、プラスチックもそれほど高価ではないため、個人的な利用や立体の試作などに使われています。

用途によってはフィギュアのように色がないと色気がない場合もあります。そのような場合は石膏を利用したものが用いられます。石膏の粉に色付きののりを吹き付けて造形します。インクジェットのプリンターのインクにのりが混ざっている感じで、平面が一つ分描けたら、その上に石膏の粉を薄くまいてまた描くという繰り返しで造形します。最終的には石膏の中からのりで固まった立体を掘り出し、さらに石膏は柔らかいので表面を薄くコーティングします。これも石膏が比較的安いので安価ですが、プリンターの大きさはかなり大きくなります。

工業的に利用するものは、プラスチックや石膏では強度が足らない場合も多いので、金属による造形が可能な 3D プリンターが用いられます。チタンによるものは、石膏と同様な方式で造形が可能なためチタンそのものは安くありませんが、一番安価に手に入るようです。ただしチタンの粉を固めるためにはのりでなく、レーザー光線で熱するため色は付けられません。それ以外の金属として、鉄、アルミニューム、銀、白金、金なども可能ですが、これらは一旦石膏などで型を作成し、そこへ溶けた金属を流し込んで作成すると言う手間のかかる方法を使用するため、時間や費用がかかります。

4.1 3D プリンターの使い方

mars には「Flash Forge Adventure III」という比較的低温で柔らかくなるプラスチックを利用した 3D プリンターが接続されており、三次元モデルさえあれば、簡単に 15cmX15cmX15cm より小さい立体を造形することができます。材質は ABS(アクリロニトリル・ブタジエン・スチレン) と PLA(ポリ乳酸) が使用可能で、現在は赤色の PLA がセットされています。PLA は ABS よりも低温で柔らかくなり、固くて脆いので加工しにくい、塗料が付きにくいと言うような特徴があります。逆に言えば ABS の方が高温に耐えて、柔軟性があるので落としても割れにくいものができます。以下に造形の手順を示します。

1. Blender で三次元モデルを作成し、上部の「ファイル」メニュー→「エクスポート」→「Stl (.stl)」と選択し、STL と呼ばれる形式で保存します。
2. 画面左下の「スタート」→「アクセサリ」→「FlashForge 3D Printer Control Software」(以下プリンター制御ソフトと呼ぶ) と選択します。
3. ロードボタンで STL 形式で保存したファイルを読み込みます。すると図 11 のようになります。
4. スライスボタンで立体モデルを輪切りにします。図 12 のような設定画面が出ます。細かく輪切りにするほどより綺麗な造形が可能ですが時間がかかります。輪切りの細かさは解像度で選択してOKボタンをクリックすると、保存先を訪ねてくるので通常そのままSaveボタンをクリックします。
5. 印刷の推定時間などを確認してから、G コードを送信ボタン[*16]をクリックします。すると「プリンターに接続する」と言うダイアログが出るので接続をクリックします。

[*16] G コードは元々工場で使われる工作機械の制御に使われるものです。工作機械では金属の塊から作りたいものを削り出して作りますが、3D プリンターでは逆に積み重ねていく感じで作るので、ヘッドの動きなどが外側からでなく内側からと言うように逆になります。

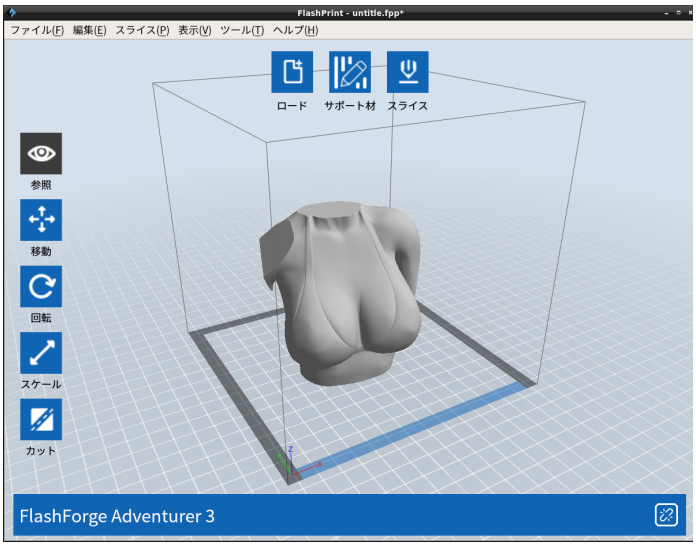

図 11　三次元モデルを読み込んだ状態

図 12　スライスの設定画面

6. 印刷が開始されると画面右下に、ヘッドなどの温度や進み具合が表示されます。プリンターの動いている様子が、図 13 のように http://mars.mgt.sugiyama-u.ac.jp/3dp.htm で見ることができます。このページはパソコンやスマホからでも見えます。
7. 印刷が終了したら、4 階の 414 室に行って造形された立体モデルを取り外します。取り外さないと次の人が出力できないので、終了したらすぐに取り外してください。立体モデルの乗っている黒いプレートの手前の部分を少し下に押すと手前にプレートを引き出すことができます。プレートを少しひねるとパリと言う感じで立体モデルが剥がれます。

複数の人が同時に 3D プリンターを使用した場合、まぜこぜの立体モデルができる事はありません。最初の人が使用中は、次の人は 3D プリンターに接続できません。造形が終わると次の人が 3D プリンターに接続できるようになります。そこで最初の人が造形された立体モデルを取り外していない場合、3D プリンターヘッドが衝突してモデルか 3D プリンターが壊れることになります。よって印刷開始の前には、必ず 3D プリンターの内部を確認してください。

図 13 3D プリンターのモニタ画面

4.2 3D プリンター使用上のヒント

インターネットで検索すればたくさんの STL 形式のファイルを見つけることが可能です。これらを頂いてきて、3D プリンターにかけることも可能ですが、大きさがプリンターの造形可能なサイズを超えている事もあります。そういう場合は、ファイルを読み込んだあとで、左側にある[スケール]ボタンで大きさを変更することができます。

この授業で使用する 3D プリンターは、立体モデルの下の方から造形します。ところが大抵の立体モデルでは下から造形すると、途中では宙に浮いてしまう部分があるものも少なくありません。例えばプリンター制御ソフトを起動して、「ファイル」メニュー→「サンプル」→「Hanako_stand.stl」を読み込んでみると、手の部分はその部分を造形する際には宙に浮いていることがわかります。そしてこのような部分があるモデルはそのままでは造形できません。スライスをする前に、[サポート材]ボタンをクリックして、[自動サポート]をクリックします。緑色の支えが表示されたら[もどる]ボタンをクリックして戻ります。このようにして立体モデルに支柱を追加します。それからスライスして造形し、できた立体モデルから支柱を物理的に切り離して完成となります。複雑な立体モデルでは内部にこの支柱が必要となり、あとでそれを除去できなくなるという事もよくあります。支柱の部分に使われたプラスチックは無駄になりますし、除去の手間を考えるとできるだけ支柱は少ない方が良いでしょう。例えば蓋のない箱の場合、蓋のない面を上に向ければ支柱は不要ですが、蓋の無い面を下にすると箱の中は支柱だらけになります。この辺りを考慮して立体モデルの作成の際に向きを決めましょう。

画面の左側にある[カット]を利用して立体モデルを分割することができます。例えば球はどのような向きにしても、床面と点で接触するので支柱が必要になります。しかし半分に切り分けて断面を下にすれば、単なる丸い山形ですから支柱は不要です。造形した後で貼り合わせる必要が生じますが、支柱を外すのより楽でしょう。

5. 拡張現実 (Argumented Reality)

5.1 拡張現実とは何か

仮想現実 (VR: Vertual Reality) というものがあります。コンピュータが作り出した仮想世界をゴーグルの形をしたディスプレイ (HMD: Head Mounted Display) で見るもので、頭や体の動きに合わせて見せる仮想世界を動かすので、本当にその世界の中に居るような感じになります。コンピュータの性能向上のおかげで本当に綺麗な仮想世界を見せることができるようになりました。一方問題点は次のとおりです。

- 綺麗な仮想世界を作るためには膨大な三次元データが必要となる。見渡す限りに存在する全ての物を用意する必要があります。全て空想の世界であれば、単純な立体で構成することも可能でしょうが、現実に近い世界を作るとなると細かいところまで忠実な三次元データを作成しなければなりません。
- 頭や体の動きと仮想世界の動きにはどうしても遅れやズレがあり、その結果長時間使用すると車酔いのような症状をもたらすことがあります。

これに対して拡張現実 (AR: Argumented Reality) は現実世界の画像の中に、コンピュータが作り出した仮想物体を表示するものです。例えば「ポケモン GO」では、スマホのカメラで写し出した現実の風景の中に、コンピュータの作り出したポケモンが登場します。周りのものや背景は全て現実のものを借りるので、仮想物体だけ用意すれば済みますので VR よりも楽です。また VR のように頭や体の動きの検出の必要もありません。もちろん全く現実と異なる世界を提供することはできません。現実世界のどこに仮想物体を出現させるかによって AR は大きく 2 つの方式に分かれます。

- **マーカーと呼ばれる図のあるところに出す方式**：マーカーのあるところを基準にして、大抵その上に仮想物体が居るように表示されます。マーカーを動かしたり、マーカーに近寄るとその動きに合わせて仮想物体が動いたり大きさが変わったりします。マーカーが必要ですが、マーカーさえあればどこにでも仮想物体を出すことができます。現在はマーカーとして普通の絵も利用可能です。例えば「日経 AR」は日経新聞記事中のグラフや写真をマーカーとしています。
- **GPS などで求めた位置を基準に仮想物体を出す方式**：「ポケモン GO」がこの方式で、おかげでポケモンを探してあちこち歩く必要があります。現在の GPS では位置の精度には数 m の誤差がありますので、出現する位置もその影響を受けます。また屋内などの GPS の電波が十分届かない場所では使えません。GPS に頼らず、スマホの向きとカメラに写った内容から地面を検出して位置を特定して仮想物体を出すことも可能になりました。こちらはスマホからの相対的な位置にマーカー無しで仮想物体を出すことができます。

この演習では大量の三次元データや HMD を必要としない AR を取り上げます。またマーカーさえあればどこでも使える方式を採用します。作成した三次元モデルを 3D プリンターを使えば現実の物にすることができましたが、AR を使えば三次元モデルが現実にあるように見せることができるようになります。

5.2 AR テストシステムの使い方

スマホのアプリで AR を検索すると多数のアプリが出てきます。それぞれ出てくる仮想物体が違います。要するに表示される物の種類だけアプリがあります。Blender で自分が作成した三次元モデルを出せるようなスマホアプリを作りかけたのですが、間に合わなかったのでブラウザで見るシステムを作りました。以下のような手順で使います。なお、単純な図形の表示で良ければ obj 形式の三次元モデルなしでも使えます。

1. Blender で三次元モデルを作成する。

マーカーとの位置関係を設定する際に、ややこしくならないように原点の位置をわかりやすい外部の点に設定します。実は obj 形式のファイルには単位情報が含まれません。よって Blender で単位をメートルにして作成することを仮定しています。

一方、3D プリンターでの出力を想定して単位を mm にして作成したモデルは、後から単位をメートルに変更しても、内部での扱いが変わらないので、次の obj 形式で保存を行うと、大きさが千倍のものとして保存されてしまいます。これを避けるには、次の手順を参考に一旦「Collada (デフォルト) (.dae)」形式で保存します。そして Blender を起動し直して最初の状態で、立方体を削除し、単位をメートルにしてから Collada 形式で保存したものを「ファイル」メニューの中の「インポート」の中の「Collada (デフォルト) (.dae)」で取り込みます。

2. 三次元モデルを obj 形式で保存する。
   a. 対象をクリックして選択する。
   b. 「ファイル」メニューの中の「エクスポート」の中の「Wave front (.obj)」を選択する。
   c. 保存場所とファイル名を指定してから OBJ をエクスポート をクリックする。これで指定した「〜.obj」ファイルと同じところに「〜.mtl」ファイルが保存される。
3. AR テストシステムへブラウザからアクセスする
   「https://mars.mgt.sugiyama-u.ac.jp/AR/」へアクセスすると図 14 のように表示されます。

図 14　AR テストシステムの最初の画面

4. 「新規登録」をクリックすると図 15 のように表示されますので、マーカーの種類などを設定し、「新規登録」をクリックします。「パスワード」、「タイトル」、「マーカーの種類」は適当に、「ターゲットの種類」は必ず「Blender などで作成した obj ファイル (要アップロード)」を選択します。さらにその下で、「ターゲットの 3D モデルの OBJ ファイル」と「ターゲットの 3D モデルの MTL ファイル:」を指定します。
5. 「新規登録」をクリックすると図 16 のように URL や QR コードが表示されるので、これらをもとにスマホやタブレットでアクセスします。なお「登録名」は 4 文字のランダムな英数字になります。URL の最後の部分がこれになるので、この登録名を忘れると再度アクセスする際に困ります。
6. 背面カメラによる映像が表示されるので、カメラをマーカーの方に向けると図 17 のように三次元モデルが出現します。映像が表示される前にカメラの使用の許可を求めてきた場合は、毎回許可しないと表示されません。

図 15　新規登録の画面

図 16　登録結果の画面

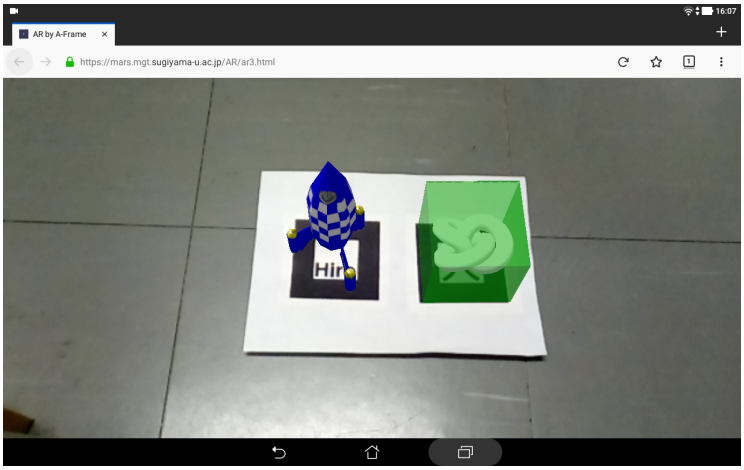

図 17　タブレットでの表示

5.3 テクスチャ付きモデルの作成方法

テクスチャとはモデルの表面の模様のことです。本来は 3 章で説明すべきことですが、Blender でテクスチャ付きモデルの作成方法をここで説明します。現在本学部にある 3D プリンターでは色付きのモデルを造形することができませんが、AR では簡単に表示することができます。むしろ色ぐらい付いていないと寂しいのと、立体モデルとしては簡単な形状だがテクスチャによっては AR で使えるものも考えられるからです。

モデルの材質を示すマテリアルには色の設定があるため、これを利用すれば色付きモデルを作成することができます。ところが同じマテリアルの部分は同じ色になるので細かい模様のようなものは表現できません。そこで色付きのモデルでは大抵マテリアルの上に模様であるテクスチャを貼り付けています。大抵のフィギュアの顔の部分も目玉などはテクスチャで描いています。テクスチャのデータとなる画像は普通の平面画像です。これを立体に貼り付けるために結構面倒な手順が必要になります。特に顔のような凹凸が色々ある場合、展開図を作成してそこに色を付けるような形になります。ここでは一番簡単な平面にテクスチャを貼り付ける方法を説明します。

1. Blender を起動し、単位をメートルに設定し、不要であれば初期設定の立方体を削除する。
2. 画面の左側の「作成」タブの中の「平面」をクリックして平面を追加する。
3. N を押して寸法設定画面を呼び出して大きさを設定する。
4. 右側の  にある 9 番目の●ボタンを選択し、下の方に出てくる新規をクリックしてマテリアルを追加する。
5. 画面中央でTabを押して「編集モード」にしてから、U を押すと「UV マッピング」のメニューが出てくるので「展開」を選択する。
6. 画面左下の 

の一番左をクリックするとメニューが出てくるので、「UV 画像エディター」を選択する。
7. 画面下部にある開くで画像ファイルを開く。画像ファイル名の隣りにあるFをクリックして固定する。
8. 画面左下の 

の一番左をクリックするとメニューが出てくるので、「3D ビュー」を選択して元の画面に戻り、下部の「編集モード」と出ているところの右をクリックしてメニューを出して「テクスチャ」を選択して画像が貼り付けられているのを確認する。画像の向きが 90 度違った場合は寸法を縦横逆に設定し直して、90 度回転させればよい。
9. Ctrl+T で三角化を実行する。すると平面に斜め線が入り、長方形が 2 つの三角形に分割される。
10. OBJ 形式で保存する。保存先は貼り付けた画像があるディレクトリ (フォルダ) にする。
11. AR システムに登録する際には、OBJ ファイル、MTL ファイル、画像ファイルの 3 つを必ず指定する。

なお、三角化するのは現在使用している AR システムが、全ての面が三角形で構成されているとしているためです。これを忘れると Blender では問題ないのですが、AR で表示する際に変になります。

演習問題

以下のようなものを AR で実現せよ。ただし誰が登録したものかわかるように、タイトルに自分の名前を追加すること。

1. 黄色の箱の中で青い箱がぐるぐる回るもの
2. 前章で作成した名前プレートをが、ゆっくり回転して裏も見えるようなもの
3. 有名絵画を実物大で見ることができるもの

6. 深層学習

この章では近年流行の AI (Artificial Inteligence: 人工知能) でよく用いられている深層学習 (Deep Learning) について学びます。

6.1 コンピュータによる問題解決

コンピュータは、何かを入れると何かを返してくれる機械、と言う捉え方をすることができます。パソコンならばキーボードやマウスで指示を入れます。スマホならばタッチして指示します。お返しは画面への表示であったり、音声だったりします。入れたものに対して、出てきたものの間には何らかの関連があります。この関連を表すものとして代表的なものがアルゴリズムです。最大公約数を求めるアルゴリズムである「ユークリッドの互除法」というのが有名ですが、このアルゴリズムに従えば必ず最大公約数を求めることができます。コンピュータの動き方を決めるプログラミングは、このアルゴリズムをコンピュータで実行できる形に書き直す作業です。

どのような問題に対しても解決するアルゴリズムが存在するのだろうか？と言う疑問はかなり昔に理論的に否定されています。つまり理論的にアルゴリズムが存在しない問題が世の中にはあります。また良いアルゴリズムが見つからない問題、一応アルゴリズムはあるのだが、ものすごく時間がかかる問題などが多数あります。とは言うものの、コンピュータに人の代わりに働いてもらおうとすると何らかのアルゴリズムが必要です。

AI はその全てがそうではありませんが、アルゴリズム無しで問題の解決をします。アルゴリズムの代わりにデータで学習して、コンピュータが自分で問題を解く方法を発見します。人の神経組織をまねた構造を作り、人の学習のまねをするという発想はかなり昔からありました。ただこの学習がなかなかうまく行かない、学習のために膨大なデータと膨大な計算量が必要ということで、長い間お蔵入りしてました。最近になってコンピュータの計算力が超強力になり、効率よく学習させる方法が見つかってきたのでこの技術が復活して、様々な分野で使われようとしています。

AI による解決の良い点は、アルゴリズムを考えなくて良いところです。問題点は、

- 適切なデータが大量に必要
- 結果がきっちり出るわけではない
- なぜその結果になるのかわからない

と言うような点です。逆にアルゴリズムの良い点は、結果がきっちり出るのと、なぜその結果になるのかが明確なところです。AI を利用したシステムは極端な言い方をすれば、独断と偏見で結果を出します。よって従来のアルゴリズムによって動いているシステムと同じように考えると危険です。いくら正しい結論が出ても、なぜそのような結論になるのかわからない場合、結論を出したのが人であれば疑いを持つ人が出ます。一方コンピュータが出したとか AI が出したとなると、信じる人がたくさん出てきます。よく SF に出てくるコンピュータに支配された未来世界は、もうすぐ実現するかもしれません。

6.2 機械学習

AI が学習するためのデータは、「教師あり」のものと「教師なし」のものに分けられます。大抵は「教師あり」です。これは正解付きデータと言っても良いでしょう。教師ありデータを学習して、正解が答えられるようになることを目指します。人のお勉強とは違って、AI は学習の際に使用したデータに対しては必ず正解を答えます。学習していないデータに対してどのくらい正しい答えを出せるかが評価のポイントになります。

「教師なし」の例としては、2012 年に Goole がインターネット上の様々な写真画像を処理した結果、猫を認識できるようになった、と言うものがあります。これは個々の画像に対して「猫である」とか「猫でない」と

いう情報を与えることなく、猫を識別できるようになりました。もちろん AI が行ったのは単なる画像の分類です。その分類された画像グループの一つを人が見たら猫ばかりだった、ということです。個々のデータに対して正解を付ける作業は大変な手間がかかることが多いので、教師なしの学習は困難ですが様々な研究が行われています。Google の作った Alpha GO[*17]と言うシステムが囲碁の世界チャンピオンに勝ったと言う話は有名なのでご存知だと思います。この世界チャンピオンに勝ったシステムは、過去の対戦記録を 3,000 万局分も学習していたそうです。ところが現在このシステムにボロ勝ちをするシステムがあります。囲碁などのゲームは勝ち負けが明確です。その性質を利用して、AI システム同士で対戦させて勝ち負けを自分らで判定して学習するようにして、過去の対戦記録なしに大量の学習を行った結果です。このようにシステムが正解を判定できるようにして実現した教師なし学習も有望です。

6.3 ニューラルネットワーク

ニューラルネットワークは、図 18 のような人などの生物の神経組織をモデル化したものです。神経細胞 (ニューロン) は、樹状突起と呼ばれる部分で他の神経細胞などからの信号を受け取り、数 mm〜数十 cm の細長い軸索で他の細胞に信号を送り出します。神経細胞では化学物質を利用して信号伝達を行いますが、これをモデルにしたニューラルネットワークでは数値を使います。神経細胞は通常複数の細胞から信号を受け取りますが、どの細胞からの信号も同じ扱いにしている訳ではありません。同様にニューラルネットワークでも、受け取った数値に適当な数を掛け算してから合計します。その結果を活性化関数というもので計算した結果を次へ送ります。ニューラルネットワークでの学習の実体は、この掛け算する数 (係数) を求めることです。

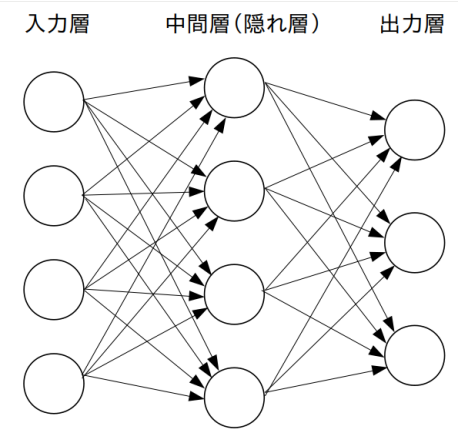

図 18 ニューラルネットワーク

図 18 では、入力層に 4 つのニューロンがありますが、この数は入力の数に対応しています。数が多い例としては、画像の入力があります。我々が目で物を見るとき、目の中の視神経が見えているものを細かく分割して脳に送っています。同様にニューラルネットワークで物体の認識をさせる場合は、画素ごとにニューロンを割り当てます。800 万画素のカメラにそのまま対応させようとすると入力層に 800 万個のニューロンが必要となりますが、さすがに多すぎるので 100×100 程度に間引いて扱うのが普通です。処理の都合から考えると少ないほどよいのですが、間引きしすぎると何の画像か人が見てもわからなくなります。人が見てもわからないものは、ニューラルネットワークで処理してもたいてい識別できません。

図 18 では、中間層は 1 段で 4 つのニューロンから構成されていますが、ここを何段にするか、ニューロンを何個にするかは決まっていません。段数や個数を増やすと処理に時間がかかり、学習がうまく行かない可能

[*17] 私が恐ろしいと思ったのは、Google はこの囲碁プログラムの開発に 900 億円かけたという話を聞いたときです。それまでに囲碁対戦プログラムの研究や開発にかけられた費用を遥かに超える額を、世界チャンピオンに勝てるめどもないのに出すと言う Google に、勝てる日本の会社はあるのだろうかと。

性が高くなりますが、段数や個数が足らないといくら学習させても正答率が上がりません。似たような事例を参考に段数と個数を決めてやってみて、だめならば少し変更してやってみるを繰り返すしかありません。

図 18 では、出力層に 3 個のニューロンがありますが、これは結果の種類の数で決まります。例えば学生さんの顔写真を入れて、可愛さのみを求めるならば、出力層はニューロンが 1 個になります。同時に美しさ、賢そうか、優しそうかも求めるならば 4 個になります。

図 18 のニューラルネットワークには矢印が 28 本 ($= 4 \times 4 + 4 \times 3$) あります。この矢印の数だけ係数があり、学習によってこれを求めなければなりません。残念ながら二次方程式のような解法はありません。とりあえず係数に適当な数値を設定してやってみて、少しずつ正しい答えに近づくように修正していくと言うような解法になります。山登りに例えると、足元の地面を見てより高い方へ向かって進んで行けば、そのうち山頂にたどり着けるだろうと言う感じです。繰り返してもあまり改善が見られなくなったところで学習終了とします。残念な結果になったら、データを増やすとか中間層の構成を変更することになります。データや中間層の構成に問題がなくても失敗することがあります。山で遭難しそうになったら、麓 (ふもと) を目指して低い方へと進むのですが、盆地にたどり着くとどちらを向いても上り坂になってしまい、進む方向がわからなくなります。同様なことがニューラルネットワークの学習にも生じることがあります。現実の山と違って地図もないので盆地にハマっているかどうかすら通常はわかりません。

確率勾配降下法と言う手法を用いて盆地にハマらないようにしています。学習用データをランダムに小分けして学習させる方法です。全員で行き先を決めると、行き先が平均化されて盆地にハマるかもしれないが、多数の小グループであればその中にはとんでもない方向へ行くグループもあるだろうし、もしかすると麓への道を見つけ出すグループが出てくるかもしれないと言う感じです。

画像認識などでより良い認識率を実現するために**畳み込みニューラルネットワーク** (CNN) と言うものも考えられました。確か猫で行われた実験だったと思いますが、生まれたときから縦縞ばかりの環境で育てたところ縦縞にしか反応しなくなり、頭の中の神経を調べたところ縦縞のみに反応する部分が見つかったが、横縞に反応する部分はなかったというものです。我々は様々な形のものを見て育ちますから、頭の中には縦縞に反応する神経だけでなく、横縞や角などの各種部分に対応する神経があるでしょう。そういうものの組み合わせとして物を認識している可能性があります。畳み込みニューラルネットワークでは画像から複数のフィルターを使って複数の特徴的な部分を探しその結果を利用して学習を行います。またどのような特徴が有効なのかは分かりませんので、並行してフィルターの内容も学習します。

英語を日本語に翻訳させる、株価の予想をさせたい、と言うような場合も図 18 のようなニューラルネットワークでは対応が難しくなります。明日の株価を予想させるために、前日の株価だけではなかなか当たりません。ある程度の幅 (例えば一週間) を持った過去の株価を元に予想した方が良いでしょう。そして、単純に過去 7 日分の株価データではだめで、株価の推移も考慮してもらわないといけません。英語や日本語もどの単語があったでは意味が取れません。同じ単語でも先頭に出てくれば大抵主語ですので語順は重要です。**再帰的ニューラルネットワーク** (RNN) では、7 日分の株価から明日の株価を予想する場合、図 18 のようなニューラルネットワークを 7 つ用意する代わりに、一つのニューラルネットワークに日にちと株価のセットのデータを与えて学習させます。このやり方ならば 100 日前の株価からだって入れられますが、あまり欲張ると学習できなくなります。

Deep Q Network (DQN) は強化学習用のニューラルネットワークです。強化学習は教師なし学習の一種と言われていますが、例えば迷路から抜け出す道筋を学習するような場合に使われます。現在位置の様子を見て、前進・後退・右へ・左へと選択をします。そのうち出口にたどり着いたらこれが正解として学習します。一番遠回りの道を選ぶようになっては困りますから、より短いコースを選択するように学習させます。囲碁の学習も同様に強化学習で行いました。現在これを車の自動運転に利用することが検討されています。車が自分の周囲を見て、ちゃんと道路の上を走るようにするためです。

6.4 様々なフレームワーク

深層学習のために様々なフレームワーク (枠組み) が作られています。フレームワークを利用することにより容易に深層学習を行うシステムを作ることができます。代表的なものとして、Google が作った Tensor Flow、Amazon が作った MxNet、Microsoft が作った CNTK、Preferred Networks が作った Chainer などがあります。どのフレームワークを使うにしろ Python と言うプログラミング言語に関する知識が必要になります。Python 自体は深層学習のためのプログラミング言語と言うものではなく、普通のプログラムが書けるものです。将来 AI システムを作るお仕事をするならば Python を勉強しておくべきですが、他のプログラミング言語とは少し書き方が違うので、将来未定の人に勧めるのはどうかなぁ、と現在は思っています。

有名な会社がそれぞれ別のフレームワークを打ち出しているのは、AI の中核とも言える深層学習での覇権を狙っているからです。多くの利用者があれば、多くの使用例が生まれ、大学などの授業で使用例が紹介され、そういう授業を受けた学生が会社で同じようにやってみようとする訳です。このテキストでは Chainer を使用することにしました。他のフレームワークが超有名会社が作っているのに対して、Preferred Networks なんて聞いたことがない会社でしょうが、実はこの会社は日本の会社です。よってトヨタ自動車など日本のメーカーや大学などとの共同開発の例などがあるようです。

6.5 深層学習モデルによる画像認識の例

株式会社明治が 40 年以上前から販売している「きのこの山」というお菓子があります。姉妹品として「たけのこの里」と言うお菓子もあります。この両者を見分けるのは幼児でも簡単な問題ですが、コンピュータではなかなか面倒なプログラムを書かなければなりませんでした。この 2 種類のお菓子を深層学習で学習させたモデルを利用して見分けるプログラム[*18]の例がリスト 1 です。

リスト 1 見分けるプログラムの例

```
1  import argparse
2  import os
3  import numpy as np
4  from PIL import Image
5
6  import chainer
7  import chainer.functions as F
8  import chainer.links as L
9  import chainer.initializers as I
10 from chainer import training
11 from chainer.training import extensions
12
13 # 畳み込みニューラルネットワークの定義
14 class CNN(chainer.Chain):
15     def __init__(self, n_units, n_out):
16         w = I.Normal(scale=0.05) # モデルパラメータの初期化
17         super(CNN, self).__init__(
18             conv1=L.Convolution2D( 3, 16, 5, 1), # 1層目の畳み込み層
19             conv2=L.Convolution2D(16, 32, 5, 1), # 2層目の畳み込み層
20             conv3=L.Convolution2D(32, 64, 5, 1), # 3層目の畳み込み層
21             l4=L.Linear(None, n_out, initialW=w), #クラス分類用
```

[*18] オリジナルは、CQ 出版社「インターフェース」2018 年 8 月号 p.41-43 の牧野浩二、西崎博光による「ラズパイでディープ・ラーニング初体験・・・画像認識」による。入力部分など一部改変している。

```
22          )
23
24      def __call__(self, x):
25          # 最大値プーリングは 2× 2,活性化関数はReLU
26          h1 = F.max_pooling_2d(F.relu(self.conv1(x)), ksize=2, stride=2)
27          h2 = F.max_pooling_2d(F.relu(self.conv2(h1)), ksize=2, stride=2)
28          h3 = F.max_pooling_2d(F.relu(self.conv3(h2)), ksize=2, stride=2)
29          # 9x9,64ch
30          return self.l4(h3)
31
32  # 画像の大きさを指定サイズに切り詰める
33  def crop(img, size):
34      w, h = img.size
35      assert w >= size or h >= size, "画像サイズが小さすぎます"
36      if w > h :
37         p = (w - h) / 2
38         box = (p,0,p+h,h)
39      else :
40         p = (h - w) / 2
41         box = (0,p,w,p+w)
42      return img.crop(box).resize((size, size))
43
44  def main():
45      # オプション処理
46      parser = argparse.ArgumentParser(description='Chainer example: MNIST')
47      parser.add_argument('--model', '-m', default='model',
48                          help='Resume the training from snapshot')
49      parser.add_argument('--unit', '-u', type=int, default=1000,
50                          help='Number of units')
51      args = parser.parse_args()
52
53      model = L.Classifier(CNN(args.unit, 2))
54      chainer.serializers.load_npz(args.model, model)
55      hantei = ['「きのこの山」','「たけのこの里」']
56
57      np.set_printoptions(precision=6, floatmode='fixed', suppress=True)
58      for imgname in [f for f in os.listdir('testdata') if ('jpg' in f)]:
59          img = Image.open(os.path.join('testdata',imgname))
60          img = np.asarray(crop(img, 100))
61          img = img.transpose(2, 0, 1)
62          x = chainer.Variable(np.array([img]).astype(np.float32))
63          y = model.predictor(x)
64          c = F.softmax(y).data.argmax()
65          s = F.softmax(y)
66          print('{:<15} は{}です。{}'.format(imgname, hantei[c], np.array(s.data)))
67
68  # main()関数を実行する
69  main()
```

このプログラムは Python と言うプログラミング言語で記述されています。さらに細かいことを言えば ver.3 の Python です。ver.2 の Python ので書かれたプログラムもよく使われているのですが、微妙に書き方

が異なります。

1-11 行 import と言う語が使われていますが、これは既にあるライブラリの読み込みを指示しています。ライブラリは Excel の関数のようなものの集まりで、Python には様々な用途のライブラリがあります。ここでは数値処理、画像処理、そして chainer などのライブラリを読み込んでいます。

14-30 行 ここで chainer で使用するニューラルネットの定義をしています。14 行目から CNN と言うクラスの指定が始まり、その中身は 15-22 行目と 24-30 行目の部分に分かれています。意外に短いですがここの書き方を理解できれば、自分の好きなニューラルネットを作ることができるようになります。Python ってすごいなぁと思われるかもしれませんが、この程度の行数で書けるのは、最初の部分で chainer のライブラリを読み込んでいるためで、Python が偉いわけではありません。プログラムの行の先頭が段々と右へ後退し、また左へもどる様子が見られると思いますが、これは Python の特徴的なところです。通常のプログラミング言語では () や {} で囲うところを、Python ではその部分を右へ後退させて示します。

33-42 行 ここで crop と言う関数を定義しています。画像処理ライブラリの助けを借りて、この関数は一つの画像データを $size \times size$ に修正します。$size$ の具体的な値は 60 行目で crop を呼び出しているところで 100 だとわかります。

44-66 行 ここでプログラムの本体となる main 関数を定義しています。関数は定義しただけでは動かないので、69 行目で呼び出しています。

46-51 行 これはプログラム起動時に指定した内容の取り込みを行っています。学習したモデルのファイル名と、unit の数を取り込んでいます。

53-54 行 ここで既に定義した CNN クラスを実体化し、指定したモデルをそこに読み込んでいます。

55 行 モデルの出力するカテゴリー (分類) の数が 2 つなので、それに対応するメッセージを 2 つここで用意しています。

58-62 行 ここで「testdata」ディレクトリ内の jpg のファイルを順番に取り出し、大きさを揃え、ニューラルネットの入力用の形式に変換しています。取り出された画像が最終的に変数 x に入ります。

63 行 ここで変数 x の内容についてニューラルネットの処理を行い結果を変数 y に入れています。

64-65 行 得られた結果から、一番大きな値となったカテゴリーの番号を変数 c に、全てのカテゴリーの値を変数 s に入れています。

66 行 変数 c の値を元にメッセージを表示しています。

結局「きのこの山」と「たけのこの里」の画像を識別するプログラムとして特有の部分は、55 行目しかありません。ここさえ直して、「かわいい女子大生」と「かわいくない女子大生」の画像を大量に用意して学習したモデルを使えば、かわいい女子大生識別プログラムの出来上がりとなります。

6.6 画像認識プログラムの実行

ここでは前節で説明したプラグラムを実際に動かしてみます。コマンドをキーボードから入力して実行すると言う一番原始的な方法を説明します。

1. 「教材フォルダ」から「deep1」のフォルダを丸ごとデスクトップにコピーします。単にドラッグすればコピーされます。
2.  をクリックしメニューを出して、「システムツール」の中の「LXTerminal」を起動します。ウィンドウが開いて「miki@mars:~$」のようなプロンプトが表示されますので、以下のようなコマンドを入力します。コマンドの最後には Enter を押します。
3. 「cd Desktop/deep1」と入力します。これでデスクトップにある deep1 ディレクトリーの中に移ります。プロンプトが少し変わります。

4. 「python3 test.py -m kt.model」と入力します。python3 が ver.3 の python を起動することを意味しています。test.py は前節で説明したプログラムが入っているファイルの名前です。-m kt.model で学習済みモデルのファイル名 (kt.model) を指定しています。すると、リスト 2 のようになります。

リスト 2 画像認識の結果

```
1 miki@mars:~$ cd Desktop/deep1
2 miki@mars:~/Desktop/deep1$ python3 test.py -m kt.model
3 take9.jpg は「たけのこの里」です。[[0.000000 1.000000]]
4 kino5.jpg は「きのこの山」です。[[0.999998 0.000002]]
5 kino1.jpg は「きのこの山」です。[[0.992926 0.007074]]
6 take13.jpg は「きのこの山」です。[[0.999270 0.000730]]
7 kino9.jpg は「きのこの山」です。[[0.999997 0.000003]]
8 kino13.jpg は「きのこの山」です。[[1.000000 0.000000]]
9 take1.jpg は「たけのこの里」です。[[0.456098 0.543902]]
10 take5.jpg は「きのこの山」です。[[1.000000 0.000000]]
```

3-10 行がプログラムの出力で、「testdata」ディレクトリーに入っていたテスト画像を認識した結果が並んでいます。行の先頭が画像ファイル名で、「たけのこの里」の画像には t で始まる名前、「きのこの山」の画像には k で始まる名前が付けてあります。ファイル名の後に認識結果があり、その後にある [ ] に囲まれた 2 つの数字は、与えられた画像をどのくらい「きのこの山」と「たけのこの里」と認識したかを示しています。残念ながら 6 行目は誤認識していますが、数字を見ても思いっきり間違えていることがわかります。

このようにして、事前に用意した学習済みモデルで事前に用意した画像の認識はできましたが、自分で用意した画像を認識させる際は、次のような手順となります。

1. スマホやタブレットで「きのこの山」や「たけのこの里」を撮影します。一つ一つ撮影するのも手間ですので、混ぜて複数個まとめて撮影しても構いません。ただ後で画像を切り分ける必要があるので重ならないように撮影します。また余分なものが入らないように、白い紙の上で、影があまり映らないようにします。この辺りの条件は学習させた画像と同じ条件にすれば良いので、白っぽいものを学習させるために背景は全て黒色にした場合は、背景は黒色にしないと認識率が下がります。
2. 撮影した写真画像をパソコンに送ります。USB ケーブルで接続して送る、メールに添付して送るなどの方法があります。
3. 写真画像を Windows に標準装備されている「ペイント」で開きます。
4. 「きのこの山」や「たけのこの里」を一つだけ  (四角形選択) で囲います。できるだけ正方形になるように「ペイント」のウインドウの一番下のところに囲いの大きさがほぼ同じ (10px ぐらいは違っても良い) になるようにします。縦と横の大きさは 100px 以上にしないと認識プログラムにはねられますのでご注意ください。そして Ctrl+C でコピーします。もう一つ「ペイント」を起動し、そちらを選択してから Ctrl+V で貼り付けます。トリミングをクリックすると貼り付けた画像以外の部分は削除されます。そしてこれを「ファイル」メニューの「名前を付けて保存」で保存します。形式は「JPEG 画像」にします。名前は「きのこの山」なのか「たけのこの里」なのか簡単に見分けられるようなものにします。
5. mars のデスクトップにある「deep1」を開き、その中の「testdata」を開いたところへ、作成した画像ファイルをコピーします。
6. LXTerminal が先程の実行した後のままであれば、↑で pytho3 で始まる行を呼び出し Enter を押せば再度認識プログラムを動かすことができます。

6.7 学習プログラムの例

ここでは「きのこの山」と「たけのこの里」を学習させる際に使用したプログラム (リスト 3) の説明をします。

リスト 3 画像学習プログラムの例

```
# -*- coding: utf-8 -*-
#from __future__ import print_function
import argparse
import os
import sys

import chainer
import chainer.functions as F
import chainer.links as L
import chainer.initializers as I
from chainer import training
from chainer.training import extensions

# 畳み込みニューラルネットワークの定義
class CNN(chainer.Chain):
    def __init__(self, n_units, n_out):
        w = I.Normal(scale=0.05) # モデルパラメータの初期化
        super(CNN, self).__init__(
            conv1=L.Convolution2D(3, 16, 5, 1, 0), # 1層目の畳み込み層(フィルタ数は 16)
            conv2=L.Convolution2D(16, 32, 5, 1, 0), # 2層目の畳み込み層(フィルタ数は 32)
            conv3=L.Convolution2D(32, 64, 5, 1, 0), # 3層目の畳み込み層(フィルタ数は 64)
            l4=L.Linear(None, n_out, initialW=w), #クラス分類用
        )

    def __call__(self, x):
        # 最大値プーリングは 2× 2,活性化関数はReLU
        h1 = F.max_pooling_2d(F.relu(self.conv1(x)), ksize=2, stride=2)
        h2 = F.max_pooling_2d(F.relu(self.conv2(h1)), ksize=2, stride=2)
        h3 = F.max_pooling_2d(F.relu(self.conv3(h2)), ksize=2, stride=2)
        # 9x9,64ch
        return self.l4(h3)

def main():
    # オプション処理
    parser = argparse.ArgumentParser(description='Chainer example: MNIST')
    parser.add_argument('--batchsize', '-b', type=int, default=100,
                        help='Number of images in each mini-batch')
    parser.add_argument('--epoch', '-e', type=int, default=20,
                        help='Number of sweeps over the dataset to train')
    parser.add_argument('--gpu', '-g', action='store_true',
                        help='GPU ID (negative value indicates CPU)')
    parser.add_argument('--model', '-m', default='test.model',
                        help='Name of model file')
    parser.add_argument('--unit', '-u', type=int, default=1000,
```

```
45                      help='Number of units')
46  args = parser.parse_args()
47
48  print('# unit: {}'.format(args.unit))
49  print('# Minibatch-size: {}'.format(args.batchsize))
50  print('# epoch: {}'.format(args.epoch))
51  print('# GPU: {}'.format(args.gpu))
52  print('')
53
54  train = []
55  label = 0
56  print('loading dataset')
57  for c in os.listdir('train'):
58      print('class: {}, class id: {}'.format(c, label))
59      d = os.path.join('train', c)
60      imgs = os.listdir(d)
61      for i in [f for f in imgs if ('jpg' in f)]:
62          train.append([os.path.join(d, i), label])
63      label += 1
64  print('')
65
66  train = chainer.datasets.LabeledImageDataset(train, '.')
67  train, test = chainer.datasets.split_dataset_random(train, args.unit)
68
69  model = L.Classifier(CNN(args.unit, 2))
70  if args.gpu:
71      gpu = 0
72      chainer.cuda.get_device(gpu).use()
73      model.to_gpu()
74  else :
75      gpu=-1
76
77  # optimizer の設定
78  optimizer = chainer.optimizers.Adam()
79  optimizer.setup(model)
80
81  # 学習データ用のイタレータ
82  train_iter = chainer.iterators.SerialIterator(train, args.batchsize)
83  # 評価データ用のイタレータ
84  test_iter = chainer.iterators.SerialIterator(test, args.batchsize,
85                                               repeat=False, shuffle=False)
86
87  updater = training.StandardUpdater(train_iter, optimizer, device=gpu)
88  trainer = training.Trainer(updater, (args.epoch, 'epoch'), out='logs')
89  trainer.extend(extensions.Evaluator(test_iter, model, device=gpu))
90
91  # 計算グラフ
92  trainer.extend(extensions.dump_graph('main/loss'))
93  trainer.extend(extensions.LogReport())
94  # エポック経過ごとの損失の値を表示したりグラフ化する
```

```
95      trainer.extend(
96          extensions.PlotReport(['main/loss', 'validation/main/loss'], 'epoch',
97                                file_name='loss.png'))
98      trainer.extend(
99          extensions.PlotReport(['main/accuracy', 'validation/main/accuracy'],
100                               'epoch', file_name='accuracy.png'))
101     trainer.extend(extensions.PrintReport(
102         ['epoch', 'main/loss', 'validation/main/loss',
103          'main/accuracy', 'validation/main/accuracy', 'elapsed_time']))
104
105     # 学習開始
106     trainer.run()
107
108     # モデルをCPU対応へ
109     if args.gpu:
110         model.to_cpu()
111     # 保存
112     print('save the trained model: {}'.format(args.model))
113     chainer.serializers.save_npz(args.model, model)
114
115 main()
```

画像認識プログラムに比べるとこちらの方が倍ぐらい長くなります。学習の過程をグラフ化する部分の他、画像認識プログラムと共通する部分もかなりあります。

2-30 行 ここの部分は同じです。つまり同じライブラリーを取り込んで、同じモデルを定義しています。

32-113 行 プログラムの本体となる main を定義しています。画像は適切な大きさに変換済みとしているので、画像認識プログラムにあった crop 関数の定義部分がありません。

34-51 行 ここでプログラム起動時に指定した設定を取り込んで表示しています。学習した結果を保存するファイル名の他、学習のやり方にかかわる設定を取り込みます。

53-63 行 「train」ディレクトリーにある学習用データを変数 train に取り込みます。学習する画像は同じカテゴリーのものは同じディレクトリーに入っていると想定しています。

68-74 行 ここで既に定義した CNN クラスを実体化し、GPU (Graphical Processing Unit) が指定されている場合は、さらに GPU 用の形式に変換しています。

77-88 行 ここでさらに学習用の設定を行っています。

91-102 行 学習がどのくらい行われたかをグラフ化したり、学習中に途中経過を表示するための設定をしています。

105 行 設定が全て終わったのでここで学習をします。

108-109 行 GPU を利用して学習した場合は、ここで GPU を使用しない形式のモデルに戻します。

111-112 行 学習済みのモデルを保存します。

こちらのプログラムも学習する画像によって変わるところはありません。この例では画像を 2 つに分類していますが、より多くの分類を行う場合も変更箇所はわずかです。

6.8 学習プログラムの実行

ここでは前節のプログラムを実際に動かしてみます。やり方は画像認識プログラムの実行とほぼ同じです。

1. 「教材フォルダ」から「deep2」のフォルダを丸ごと mars のデスクトップにコピーします。

2. メニューから「LXTerminal」を起動します。
3.「cd Desktop/deep2」を入力して deep2 ディレクトリーの中に入ります。
4.「python3 learn.py -u 11 -e 120 -m kt.model」と入力して学習プログラムを実行します。するとリスト 4 のようになります。

リスト 4 画像の学習

```
miki@mars:~$ cd Desktop/deep2
miki@mars:~/Desktop/deep2$ python3 learn.py -u 11 -e 120 -m kt.model
# unit: 11
# Minibatch-size: 100
# epoch: 120
# GPU: False

loading dataset
class: kinoko, class id: 0
class: takenoko, class id: 1

epoch  main/loss  validation/main/loss  main/accuracy  validation/main/accuracy
    elapsed_time
9      18.1198    130.51                0.54           0.615385
         3.04764
18     190.31     66.4328               0.36           0.615385
         6.43201
27     87.8045    9.37778               0.36           0.692308
         9.64111
36     0.485521   30.7086               0.72           0.461538
         13.0165
45     15.6096    22.4464               0.65           0.461538
         16.7924
54     7.74122    5.34388               0.72           0.692308
         20.8998
63     1.12534e-06  18.2803               1              0.615385
          24.2213
72     14.5505    6.39744               0.37           0.846154
          27.7738
81     0.00121356  9.0284                1              0.538462
          31.2788
90     0.0035279  19.4613               1              0.461538
          34.6088
100    1.80278    24.5368               0.91           0.461538
          38.4735
109    1.27489    20.0871               0.82           0.384615
          42.2494
118    0          19.6306               1              0.461538
          46.1012
127    0          20.4532               1              0.461538
          50.3659
save the trained model: kt.model
```

約 50 秒でこれらの結果が得られます。なお LXTerminal の画面の幅を広げないとこのように途中で改行が入りますので、ちょっと見にくい表示になります。「loss」(損失、エラーの量) の値が小さく、

「accuracy」(正確さ) の値が 1 に近いほど良いのですが、この例のように途中でまた悪い方へ変わることもあります。名称に validation がついた方は、学習用データを別の形で取り出した評価用データで測定した結果です。この例の場合、最終的には学習データに対しては 100% 正確になっていますが、評価用データに対しては 46% の正確さですから外している方が多くなっています。この学習の様子をグラフ化したものが「logs」ディレクトリーの中にあります。例えば「accuracy.png」をダブルクリックするとグラフが表示され、学習用データでの正確さは上がっていますが、評価用の方はあまり上がっていない様子が見られると思います。

5. 「python3 test.py -m kt.model」で学習したモデルで画像の認識を行うことができます。結果はリスト 5 のようになります。

リスト 5 画像学習の結果の確認

```
miki@mars:~/Desktop/deep2$ python3 test.py -m kt.model
take9.jpg       は「たけのこの里」です。[[0.000000 1.000000]]
kino5.jpg       は「たけのこの里」です。[[0.001053 0.998947]]
kino1.jpg       は「たけのこの里」です。[[0.000000 1.000000]]
take13.jpg      は「きのこの山」です。[[1.000000 0.000000]]
kino9.jpg       は「たけのこの里」です。[[0.000000 1.000000]]
kino13.jpg      は「たけのこの里」です。[[0.000000 1.000000]]
take1.jpg       は「きのこの山」です。[[0.999999 0.000001]]
take5.jpg       は「きのこの山」です。[[0.869498 0.130502]]
```

残念な結果になっていますが、学習データが少ないのでしょう。ランダムで与えた初期値の影響も大きく、同じデータかつ同じパラメタで学習しても、全て「たけのこの里」や全て「きのこの山」と言う結果になることもあります。

矢印キーで前に入力したコマンドを呼び出して一部修正することもできます。学習の際の「-u 11 -e 120」の数字を変更して、再度学習をさせることができます。u の値はある程度大きい方が良いようですが、学習データの数より大きな値にすることができません。e の値を大きくすると学習の回数を増やすことができます。回数が足らないとだめですが、増やせば良くなるとは限りません。**過学習**と言う状態になると、学習用データに適応しすぎて、それ以外のデータに対しての応答がだめになってしまいます。また回数を増やせばそれに比例して時間もかかるようになります。

学習にかかる時間を短縮するために GPU などが用いられます。GPU は本来動画の処理 (特に動きの激しいリアルなゲームなど) のために作られたものですが、比較的安価に入手できるために最近のスーパーコンピュータでも使われています。さらに学習用であれば、それほど高精度の計算は必要ないことも判ってきたので、専用の処理装置も使われるようになっています。mars には Nvidia 社製の GeForce GTX 1080 Ti と言う GPU が付いています。その内部には 3,584 個の演算器があり、全体で 11.3TFLOPS[*19]の計算ができます。これを学習の際に使う場合は、「-g」を追加します。先程は約 50 秒かかりましたが、5 秒程度で終わるようになります。

自分で用意した学習データを用いて学習させる場合には、「moto」ディレクトリの中にある「kinoko」と「takenoko」ディレクトリに画像を入れます。既に入っている画像はそのままにしても良いし、置き換えても良いです。そして「python3 crop.py -i moto -o train」を実行すると、「moto」ディレクトリに入っていた画像ファイルが、100×100 の大きさに整形されて「train」ディレクトリに入ります。既に入っていたファイルは削除されますのでご注意ください。そして learn.py を実行すれば学習が行われます。

[*19] FLOPS は 1 秒間に何回小数点付きの数の計算ができるかというもの。11.3TFLOPS は 1 秒間に 11.3 兆回計算できることになります。

6.9　学習用データを増やす

とにかく学習用データを増やさないと精度良く認識することができません。一方学習用データを大量に用意するのは費用や手間がかかることは確かで、いかに効率よく用意するかが重要になります。効率の良い収集方法はデータによって異なります。消費者の行動データのようなものは、スーパーやコンビニのレジの情報からも得られます。これらは既に商品発注や在庫管理システムで利用されていますので、関係者であれば容易に入手することができるでしょう。一部のコンビニで行われていた、レジで入力したお客さんの年齢のデータもあれば、より興味深い結果が得られるかもしれません。ただ個人情報保護には注意しなければなりません。かわいい女子大生の顔写真を集めるために、現マネ棟の入り口に密かにカメラを設置して撮影などすると問題になるでしょう。ただ密かに撮影して、分類して、学習させて、学習結果だけ残した場合、どのような画像を用いたかは分かりません。つまり、いつの間にか自分の個人情報を学習データとして使われても、本人は気が付けないので注意が必要です。と言われても、対策は AI の得意な企業にはできるだけ近寄らない、でしょうか？

画像認識の場合、認識する対象は様々な向きや大きさであることが普通です。深層学習が使われる以前の画像認識では、認識処理に入る前の前処理として、撮影した画像に映るものの向きや大きさなどを揃えていました。深層学習を使う際にも前処理をした方が効果的ですが、学習データに逆の処理を行うことがよくあります。つまり撮影した学習用データを回転させたり、縮小したものも学習させることにより、回転や縮小に強くなりますし、このようなデータの水増しは機械的にできるので手間もかかりません。学習用データの機械的な収集と効率的な学習データの水増しは、深層学習の精度を上げるための大事なポイントです。

6.10　水増し学習プログラムの実行

ここでは実際に水増ししたデータをもとに学習プログラムを実行し、その効果を確認します。

1. 「教材フォルダ」から「deep3」のフォルダを丸ごと mars のデスクトップにコピーします。
2. メニューから「LXTerminal」を起動します。
3. 「cd Desktop/deep3」と入力して deep3 ディレクトリーの中に入ります。
4. 「python3 cropmod.py -i moto -o train」と入力すると moto ディレクトリーの中の画像ファイルを、100×100 の大きさに整形した後で、反転、回転、80% に縮小したものを生成して、各々別のファイルとして train でディレクトリーに保存します。moto ディレクトリーにはきのことたけのこの画像が各々 12 個ずつ入っていますが、これが各々 300 個の画像になります。
5. 例えば「python3 learn.py -u 50-e 100 -m kt.model」と入力して学習プログラムを実行します。時間がかかるようであれば、GPU も使いましょう。そして「python3 test.py -m kt.model」でできたモデルがちゃんと認識するか確認することができます。「-u」や「-e」の値を適切に設定するとリスト 6 のような認識結果も得られます。

リスト 6　水増し画像学習の結果

```
1 miki@mars:~/Desktop/deep3$ python3 test.py -m kt.model
2 take9.jpg      は「たけのこの里」です。[[0.000000 1.000000]]
3 kino5.jpg      は「きのこの山」です。[[1.000000 0.000000]]
4 kino1.jpg      は「きのこの山」です。[[1.000000 0.000000]]
5 take13.jpg     は「たけのこの里」です。[[0.000000 1.000000]]
6 kino9.jpg      は「きのこの山」です。[[1.000000 0.000000]]
7 kino13.jpg     は「きのこの山」です。[[1.000000 0.000000]]
8 take1.jpg      は「たけのこの里」です。[[0.000000 1.000000]]
9 take5.jpg      は「たけのこの里」です。[[0.000000 1.000000]]
```

7. 動画プレゼン

かつて動画の処理はコンピュータにとって鬼門でした。動画のデータ量は文字データや音声データと違って桁違いに大きく、数秒の動画すらメモリーを溢れさせ、数分の動画でディスクがいっぱいになりました。FFT (高速フーリエ変換) アルゴリズムが発見されて、音声データを高速に分解して成分が求められるようになりました。そして成分を適当に間引くことにより、音声データを大幅に圧縮することができるようになり、iPod などの携帯音楽プレーヤーが実現し、圧縮なしの CD (コンパクト・ディスク) は次第に消えていこうとしています。それでも動画となると、FFT などを使用してもなかなか追いつけない状況が続いています。どの程度の品質の動画を求めるかにもよりますが、高品質の動画はパソコンレベルでは圧縮などに結構時間がかかります。大容量のディスクが安価になりましたが、それでも圧縮しない状態の長時間の動画はなかなか保存できません。

一方、文字データ、画像データ、音声データ、動画データなどを比べてみると、他のデータを全て取り込むことができる動画データはアピール力があります。百聞は一見にしかずとも言います。適切な画像データは文字や音声では伝えきれないものを一瞬で伝えます。動画はさらにその画像が動くわけですから強力ですが、その力を活かすのはなかなか難しいところがあります。お金のかかっている動画と言えば映画があります。数十億円以上の費用がかかった映画が毎年いくつか登場します。しかし数年後にも話題になる映画は、一番費用がかかった映画ではないことも少なくありません。その時代に合った内容の映画でないと人気は出ませんし、普遍的なテーマでなければすぐに忘れ去られてしまいます。

プレゼンテーションは、発表などのことですが、様々な場面で用いられて、何らかの情報を聞き手に伝えるものです。相手がこちらが伝えたい情報をぜひ知りたいと言うのであれば、かなり雑なプレゼンテーションでも構わないでしょう。一方相手側がこちらの情報にあまり興味を持っていない場合、なかなか厳しい状況になります。文章を書いても読んでもらえない、話しかけても聞いてもらえない、そういう状況でも自分の会社の新製品について是非知ってもらいたいと言うようなことがよくあります。そこで最強の動画を使うわけですが、しつこくやると嫌われるので TV の CM などは数十秒で終わります。

最初に述べたように動画の処理は現在のパソコンで扱うにしても重たい処理になります。新海誠監督のように 25 分のアニメ映画を 7 ヶ月かけて一人で作ってしまった人もいますが、ここでは適当な動画があればそれも取り込むが、基本は静止画を見せるという PowerPoint によるプレゼンテーションとあまり変わらないものを取り上げます。元々は Windows に付属していた「Windows Live Movie Maker]」を利用した動画作成を考えていたのですが、大変残念なことに Windows 8 以降はこれは付属しなくなりました。このソフトは動画編集ソフトとしてはかなり低レベルですが、その分使いやすいと言う特徴がありました。ここで取り上げる「Openshot Video Editor」は著名なソフトではありません。Linux だけでなく Windows でも無料で使えるということで選択しました。やはり市販の動画編集ソフトは高機能で良いと思います。有名なソフトは大抵 30 日間の試用が無料[*20]でできるようになっているので、ちょっと使わせてもらって自分の技量やパソコンの能力に合っているかどうか確認しましょう。

動画によるプレゼンテーションの良いところは、

- PowerPoint の発表では発表者が必要ですが、動画の再生には発表者は不要です。よって繰り返し発表するような場合でも負担がありません。
- YouTube などの世界に公開する場があります。YouTube に登録すれば、無料で世界中の人に動画を届けることができます。また見る方もいつでもどこからでも見ることができます。

などがあります。

[*20] 私の長男は結婚式の披露宴で上映する二人の生い立ち動画を、adobe premiere elements と言うソフトで作成しましたが、30 日以内に作成して無料で済ませました。結婚式場にこの動画の作成を依頼すると 20 万円ぐらいだったと思います。

7.1　動画作成の手順

レポート作成でも元となる資料の収集は必要になります。動画を作るとなると多種多様なデータを集める必要があります。漠然と集めていると無駄が多いので、全体の構成を考えてから集めましょう。

7.1.1　動画の長さ

全体の構成を考える際に一番の制約となるのは動画の長さです。TV の短い CM は 15 秒ですし、TikTok は 60 秒以内です。YouTube に投稿する動画であればもう少し余裕がありますが、長い動画は途中までしか見てもらえない可能性が高くなります。一発芸で終わりならともかく、大抵は色々伝えたいのだが時間が足りない！と言う問題で苦労します。PowerPoint ではよく細かい字で詰め込んだスライドが見られます。動画で同じことをすると多分誰も読んでくれません。読み上げる音声を入れると時間が足りません。とにかく絞り込むしかありません。何も考えずに動画を作成していると、どんどん長くなります。動画を見てくれる人の我慢の限界を考えてまず動画の長さを決めます。

7.1.2　動画の画面の大きさ

PowerPoint ではデザインにこだわりのある人以外は気にしないところですが、プレゼンテーションを行う際の画面の大きさの設定があります。もちろん絶対的な大きさはプロジェクターとスクリーンの距離によって変わります。PowerPoint で設定できるのは**縦横比**です。「4:3」と「16:9」が選択できるようになっています。横長かかなり横長の選択で、昔ながらの TV の画面が前者で、新しい特に大型 TV やパソコンも最近のものは後者です。縦横比が合わない場合、切り詰められたり足らない部分が黒い帯になったりします。動画でも基本的にこのどちらかの縦横比を選択することになります。途中で縦横比の変更は可能ですが、デザインの面でやり直しになる可能性もあるので、最初によく考えて決めます。

近年の動画はスマホで再生される機会も多くなっています。その場合は「16:9」で良いのですが、縦向きの使うのが普通のスマホでいつもの向きのままで再生できるようにとするならば「9:16」で作成します。

さらに動画では縦横のドットの数も決めます。ドットの数が多ければ画面が綺麗になりますが、ファイルが大きくなって、通信環境によっては動画の再生に支障が生じることがあります。逆にドットの数が少なければ画面が荒くなりますが、ファイルが小さくなり、動画の再生や保存の際に有利となります。通常ドット数は、最終的な保存の際に指定する形になるので、いくつか保存して試してみれば良いでしょう。

7.1.3　説明文章

百聞は一見に敷かずとも言いますが、見ただけでは何を言いたいのかわからない画像や動画もいっぱいあります。子供が走っている動画を見ただけで、「これは平成 25 年の秋の幼稚園の運動会のリレーだ」なんてわかる人はいません。最低限の説明を文字で行うとして、それを文字を表示して行うか、音声で入れるかの 2 つが考えられます。映画で言えば字幕でするか吹き替えでやるかです。ここでは文字で表示する方法を採用します。音声でやる場合の問題は、誰の音声を使うかがまずあります。アナウンサーのように明瞭に話せる人はなかなか居ません。自分でやるのは恥ずかしいとか、録音する際に周囲の雑音が入らないようにするのは意外と難しいこともあります。YouTube などでは合成音声を使用している例も見られます。恥ずかしいとか周囲の雑音の混入などの問題はありませんが、あまり聞きやすい音声ではない[*21]こともあって、説明文も同時に表示されています。

文字を表示する際には、一度に全てを表示するのではなく、読むスピードを考えて順番に出します。PowerPoint であれば、アニメーションの機能を利用して説明に合わせて順次表示するのが良いとされています。これは一度に全て出してしまうと、見ている側が先読みをしてしまい、せっかくの発表を聞いてもらえな

[*21] ちゃんとお金を出せば、もう少しまともな合成音声も可能ですが、そういう例を見たことがありません。

くなるからです。動画の場合は、読み上げ音声なしであれば、先読みされて構わないのですが、全てを一度に出して、一度に消すと読むのが遅い人は着いていけません。順次表示して順次消えていくようにすると、読むのが遅い人もある程度飛ばして見てくれます。とにかく文字数が多いと早く読めません。映画の字幕や TV のニュースの項目の書き方を思い出してできるだけ短くしましょう。

日本人の文字を読む速さは、400〜600 文字/分と言われているそうです[*22]。誰でも間に合う、画像を見てもらう時間も必要と考えると、400〜600 文字/分はきついでしょう。

7.1.4 BGM

説明音声なしになると無音の動画になります。ビデオカメラで撮影した動画の編集による動画であれば、何らかの音声が入っているのでそれを利用すれば良いでしょう。ただいくつかの音声付きの動画を合わせる場合、音量を揃えないとびっくり動画になりやすいので注意します。

画像と文字から作成した動画には元からの音声がありません。無音の動画も寂しいので BGM (Back Ground Music) を付けることも検討しましょう。ただ音楽にも著作権があります。自分で BGM を作曲できる人はそれほど居ませんので、他の人の作曲した音楽を使うことになります。TikTok ではこの著作権の問題をクリアーした上で多くの曲を提供しています。また YouTube は「YouTube オーディオライブラリ[*23]」というものがあり、ここにある曲は自由に使えるようですが、youTube の外では使えません。

Google で「著作権フリー bgm」をキーワードに検索すると数多くの著作権フリーの BGM を扱ったサイトがたくさん出てきます。通常無料で使えるものが多いのですが、使用した動画の最後に誰の曲なのか明記するのが条件とか、商用はだめ、アダルトはだめとかの条件がついていますのでよく確認しましょう。このような曲をなぜ無料で公開しているのか？と言う疑問も出ると思います。一つは作曲者の宣伝のためと言うのがあるでしょう。実際に作曲者に有料で別の曲の作成を依頼できるようなサイトもあります。以前このようなサイトを利用した際に困ったのは、曲がありすぎてどれにしたら良いのかわからない、という点です。これから作る動画にふさわしい BGM とはどんな曲でしょうか？また曲を終わりまで聞かないと判断しにくいところも多数の曲から選ぶ際に困る点です。

7.1.5 **画像**

写真画像にも著作権や肖像権など様々な権利が絡みます。自分の会社のものならば大丈夫とも限りません。椙山女学園では、赤丸に木の「学園章」については申請を出してもなかなか使用許可が出ませんが、緑の Sugiyama の「コミュニケーションマーク」は勝手に使っても構わないようです。なぜそうなっているのかよく分かりませんが、同様の訳のわからないこだわりは、どこの組織でもありそうなので注意しましょう。

無料で使えるイラスト類は Google で検索するといくらでも出てきます。ものの説明用の写真などは Wikipedia を利用すると説明用文章の元まで同時に手に入るかもしれません。「コンピュータと情報 I」のテキストにも出てくる「クリエイティブコモンズ[*24]」などで探すのも良いかもしれません。

7.2 **動画作成の例**

ここでは「OpenShot Video Editor」を使用して動画を作成する例を説明します。「現代マネジメント学部の歴史」動画用の素材を用意しましたのでそれを使います。「教材フォルダ」から「movie1」のフォルダをドラッグして mars のデスクトップにコピーしてください。「OpenShot Video Editor」の起動は、  をクリックしメニューを出して、「サウンドとビデオ」の中の「OpenShot Video Editor」をクリックします。すると図

[*22] 「読書速度ハカルくん」(https://www.sokunousokudoku.net/hakarukun/) で測定したところ、私は 1286 文字/分で 2/2 問正解で平均よりやや速いという判定でした。ちょっと焦って読んだので正解できて良かった。

[*23] https://www.youtube.com/user/AudioLibraryJP

[*24] https://search.creativecommons.org/ただし検索語は英語で入れる必要がある。

19 のような画面が表示されます。

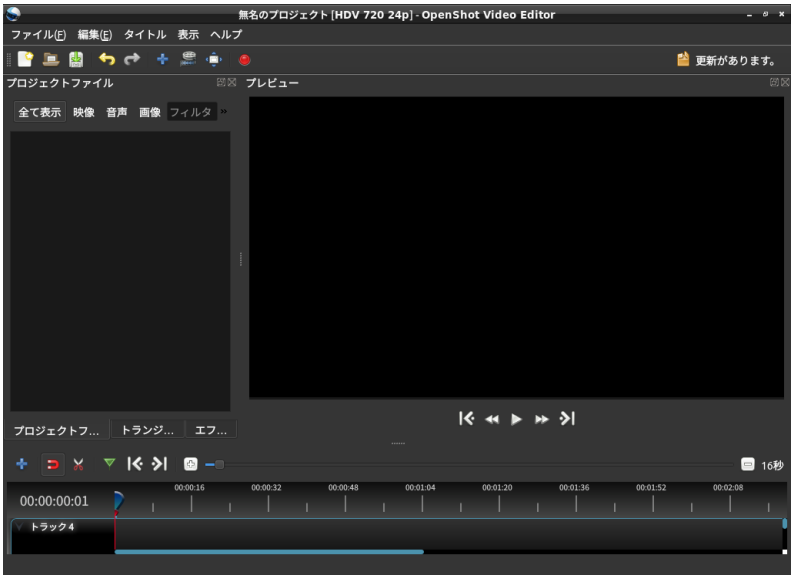

図 19　OpenShot Video Editor の画面

1. 「ファイル」メニューの「ビデオ形式の選択」で動画のサイズを設定することができます。「プロファイル」のところで適当なものを選択します。「FPS」は 1 秒間にいくつの画面を表示するかで、数字が大きいほど滑らかな動きを見せることができますが、ファイルサイズが大きくなり、再生の際にそれなりの処理能力も必要になります。とりあえず初期設定されている「HDV 720 24P (1290x720)」のままで話を勧めます。
2. とりあえずプロジェクトを保存しましょう。「ファイル」メニューの中の「プロジェクトを別名で保存」で「Desktop」の「movie1」の中に「現代マネジメント学部の歴史.osp」という名前で保存します。この「OpenShot Video Editor」はかつてよく落ちるソフトだったそうなので、こまめに上書き保存をしましょう。
3. 「タイトル」メニューで「タイトル」を選択すると静止画のタイトル、「動画タイトル」を選択すると動画のタイトルを挿入することができます。「動画タイトル」では、Blender を用いて動くタイトルを作ってくれるのですが、現在のところ英字しか扱えないのが残念です。どちらを選択してもさらにテンプレートを選択できるようなっています。「タイトル」を選択しさらに「Oval 2」を選択すると図 20 のようになります。

   「ファイル名」に「タイトル」、「行 1」に「現代マネジメント学部の歴史」、「行 2」に学籍番号と名前を入れましょう。なお、文字入力する内容は、「movie1」の中にある「文書.txt」ファイルの中に入れてあるので、このファイルを予めダブルクリックして開いて、コピー＆ペーストしても良いでしょう。
   入力が終わったら保存をクリックします。すると OpenShot の画面の左側のところにこのタイトルのアイコンが表示されます。
4. タイトルのアイコンを OpenShot の画面の下側にある「トラック 3」にドラッグします。トラックにドラッグしたものが動画のもとになります。左から右へと表示されるので、最初に表示されるタイトルのアイコンは、トラックの一番左側にドラッグしてください。トラックが複数あるのは、上にあるトラックの内容が下にあるトラックの内容を上書きするようになっているからです。つまり下のトラックに背景となるようなものを入れます。

   OpenShot の画面の右上に大きくタイトルが表示されます。ここにはトラックの赤線のあるところの動画が表示されます。赤線はその上方にある  をドラッグすると横に移動することができます。また、

図 20 Oval 2 の設定画面

をクリックすると動画の再生が始まります。トラックの横軸は時間ですが、少しメモリの間隔が小さい場合は、トラックの上にある のスライダを動かして調整ができます。

5. トラック上にあるタイトルのアイコンを右クリックするとメニューが表示されます。ここで「フェード」→「クリップ開始端」→「フェードイン (高速)」を選択すると最初ふわっと出てくるようになります。同じくメニューで「フェード」→「クリップ終端」→「フェードアウト (遅い)」を選択すると最後にふわっと消えるようになります。
6. メニューバーのすぐ下のところにボタンが並んでいます。 をクリックすると、このプロジェクトに取り込む画像、動画、音声などを選択することができます。「エンドロール.svg」、「パステルハウス.mp3」、「建物.png」、「説明 2.svg」、「説明 3.svg」を必要に応じて取り込んでください。取り込めばトラックにドラッグすることができるようになります。
7. 「建物.png」を取り込んで、「トラック 3」のタイトルの隣にドラッグします。トラックの赤線をその上に持っていくと、建物の写真が中央に表示されているのがわかります。説明文を右に流すためにこれを左に寄せるには、右クリックしてメニューを出して「Properties」を選択し、トラックの横に出てきた設定の中の「重力」の右の「中央」を右クリックして「左」を選択します。
8. 建物の写真が突然現れると驚かせてしまうので、フェードインではない効果を追加してみます。画面左側の中頃にある「トランジション」のタブをクリックすると、様々な遷移効果が出てきます。一番左上にある「円中から外」を「トラック 3」の「建物.png」の左端にドラッグして重ねると、中心から段々明るくなって建物が見えてくるような感じになります。やってみて思うような効果でなかったら右クリックして「トランジションの削除」を選択すれば効果を外すことができます。
9. 最初の説明文をタイトルを用いて作成します。他のソフトで画像として作成した説明文でも良いです。「タイトル」メニューで「タイトル」を選択し、最後の方にある「説明中 12」と言うものを選択します。これで最大 12 行までの説明を表示することができます。似た名前のものとして「説明右 12」と「説明左 12」がありますが、これは入力した説明が右揃えになるか、左揃えになるかの違いです。「説明中 12」では中央揃えになります。そしてファイル名を「説明 1」にし、説明文を 1 行毎に「行 1」から入れていきます。余った「行 7」以降は中身を消して空にします。そして保存をクリックします。

   左側に「説明 1.svg」と言うアイコンが表示されるので、それを右クリックしてメニューで「Edit Title」を選択すると内容を再修正できます。説明文が動画の中央でなく右端や左端に出て欲しい場合や、文字の大きさを変更する場合はここで修正します。一番下の「高度な設定」のところの「外部ソフトで編集」をクリックすると「Inkscape」と言う作図ソフトが起動されます。初回に「COnvert Legacy Inkscape file」というのが表示されたらOKをクリックします。
10. 「Inkscape」もお絵描きソフトの一つですが、既に登場した「Pinta」や Winodws に付属する「ペイン

ト」と違って SVG 形式の画像を扱うことができます。SVG 形式は画像を点の集まりではなく、図形の集まりとして扱う形式で、拡大や縮小をしても綺麗という特徴がありますが、ちょっと使いにくいものです。既にあるものを選択する際には、左側で  をクリックしてから、文字に関する修正をするならば、左側で  をクリックしてから行います。

11.  をクリックしてから白い文字の説明文をクリックして選択し、ドラッグして説明文の位置を右端に変更します。  をクリックしてから、説明文を選択し文字の大きさや色を変更することができます。ただあまり色々変更すると、後で利用できなくなったことがあるのでご注意ください。
12. 説明文の修正が終わったら、「Inkscape」の「ファイル」メニューで「保存」を選択してから「Inkscape」を終了します。タイトルの編集画面に戻るので、さらに保存をクリックしてタイトルの編集を終わります。
13. 「建物.svg」のトランジションの設定の終わったところに合わせて、「トラック 4」に「説明 1.svg」をドラッグします。表示時間が 10 秒では少し短いので、左のプロパティのところの「終了」をクリックして 10.00 となっているところを 15.00 に変更します。それから「トラック 4」上の「説明 1.svg」を右クリックして、「アニメーション表示」→「クリップ全体」→「端から端へ」→「下から上へ」を選択します。これで説明文が 15 秒かけて下から上へ動くようになります。ただこれだけでは動くスピードが一定でないので変です。左のプロパティのところの「Y 座標」のところにある右上がりの曲線の絵を右クリックすると、メニューが出るので「リニア」を選択します。
14. 「説明 2.svg」と「説明 3.svg」はできたものがあるので、  で取り込み、「トラック 4」へドラッグしてください。「説明 1.svg」の説明文が上に消えた時点へ、トラックの上の赤線を持っていき、その赤線に左端が一致するようにドラッグします。そうすると「説明 1.svg」と「説明 2.svg」が一部重なりますが、説明の一団が消えると次が続いて現れるようになります。

    「説明 2.svg」に対して「説明 1.svg」と同様の設定を行った上で、同様に「説明 3.svg」を追加してください。
15. 「説明 3.svg」の終端と同じところまで「建物.svg」の終端をドラッグして伸ばします。それから「建物.svg」に「フェードアウト (遅い)」の設定をします。このような終端の設定は、その開始位置が終端との相対時間差ではなく、始端からの絶対的な時間差で行われるため、設定をしてから終端が後になっても設定が取り残されてしまいます。よって長さが確定してから設定をしてください。
16. 最後に「エンドロール.svg」を取り込んで、「建物.svg」の後にドラッグします。適当に開始端の効果を設定し、終端は「フェードアウト (遅い)」にしましょう。

図 21　トラックの設定部分

17. BGM として「パステルハウス.mp3」を取り込んで「トラック 2」へドラッグするとエンドロールの終わりより先まで続くので、右端をドラッグして長さを揃えるか、トラックの上にある  をクリックしてから、「パステルハウス.mp3」の切りたいところをクリックすると、「パステルハウス.mp3」が 2 つに分かれます。そして後半の方を削除すれば切り詰めたことになります。長さの調整ができたところで、「トラック 2」上の「パステルハウス.mp3」を右クリックして「ボリューム」→「クリップ開始端」→「フェードイン (高速)」と「ボリューム」→「クリップ終端」→「フェードアウト (遅い)」を設定しま

す。「フェード」を設定しても音量は変わらないので注意しましょう。以上の操作を行うと大体図 21 のようになります。

18. プロジェクトの上書き保存をしてから、メニューバーのすぐ下の  をクリックすると動画ファイルの作成ができます。図 22 のような設定画面が出てくるので、「ターゲット」、「映像のプロファイル」、「品質」などを選択してから「動画を書き出し」をクリックします。「ターゲット」として「MP4 (h.264)」は割と標準的な設定ですが、mars の Firefox では音が出ません。「OGG (theora/vorbis)」は新しい規格で動画ファイルの大きさが小さくなります。「映像のプロファイル」は OpenShot の初期設定に合わせるのならば、「HDV 720 24p (1280x720)」にしましょう。「品質」をあげると動画ファイルのサイズが大きくなります。

図 22 動画の設定画面

8. スマートスピーカー

音楽などを聴くために、電気信号を音声信号に変換するものがスピーカーです。ラジオ用のスピーカーとレコード再生用のスピーカーのどちらが古いのかは分かりませんが、戦前からスピーカーが家庭にありました。再生する内容が、ラジオ電波で提供されるのか、レコードから提供されるかの違いがありましたが、物自体は同じで同じ原理のものが現在でも使われています。やがてレコードが CD に代わり、さらに電波の代わりにインターネット経由で再生するデータが送られてくるようになりましたが、これだけでは「スマート」とは言えません。音声による入力とそれによって制御できるものが「スマートスピーカー」となります。ただ単純に音声で音量を変えることができると言うレベルでは、スマートとは言い難いので、インターネット経由でサーバーに音声データを送って何かできるのが必要なようです。

図 23　Amazon Echo

図 24　Google Home

図 25　Clove WAVE

代表的なものとして、Amazon の「Amazon Echo」(図 23)、Google の「Google Home」(図 24)、LINE の「Clove WAVE」(図 25) などがありますが、他にも Sony など様々な会社から売られています。値段を見るとこの三社のものは一万円程度と比較的安価ですが、他のものは数万円します。これはどこで儲けようとしているかの違いです。この三社の製品については、「Amazon Music」、「Google Play Music」、「LINE MUSIC」などで音楽配信を行っており、ここで稼ぐことができます。さらに音声入力によって何かサービスを提供し稼いでいるので、「スマートスピーカー」自体で利益を挙げる必要はあまりないようです。一方 Sony などの会社では高い品質の音楽の再生を目指していますが、利用者がいくら音楽の再生をしても儲からないので、「スマートスピーカー」の販売の時点で費用を回収しなければならないので、値段が高くなります。Amazon や Google はサービスで儲けられるので、他社の「スマートスピーカー」でもスマートスピーカー用サービスが使えるようにしています。そのために Bose 社の製品のように Amazon と Google の両方のサービスが利用できるものまであります。

8.1 スマートスピーカーの構造

機能の面から「スマートスピーカー」はほぼ同じような内部構造を持つことが考えられます。

- スピーカーなどの音声再生装置：名称からしてスピーカーの部分を省略する訳には行かないでしょう。低音を出す楽器は大きく、高音を出す楽器は小さいように、本来出す音の高さと大きさは関連があります。大きな低い音を出すためには、ゆっくり大きく振動する必要がありますが、小さなスピーカーではそれは無理です。あらゆる高さの音を出すスピーカーは低音を出すためにはある程度の大きさが必要となります。またステレオ効果を出すためには、スピーカーが 2 つ必要となります。これらの面から考えると、小さな「スマートスピーカー」単体で良い音を再生するのは技術的に難しいことになります。

  また、次のマイクで再生する音を拾ってしまうので、スピーカーの部分は本体から切り離したほうが

良いのではないかと思います。内部の仕切りなどで、できるだけスピーカーからの音がマイクに伝わらないようにしていますが、本体が小さいのでなかなかうまく行きません。

- マイクなどの音声入力装置：音声による指示を受け取るためのマイクが必要です。どこから声をかけても大丈夫なように、通常複数のマイクを内蔵しています。複数のマイクで取り込んだ複数の音声信号から、共通する部分を除くことにより、本体のスピーカーからの音をある程度削減することもできます。マイクの感度が低いと、大きな声で指示を出さねばなりません。マイクの感度が高いと、テレビの音[*25]や隣の部屋の話し声に反応してしまう可能性が高くなります。
- 音声認識装置：最初の呼びかけの認識は「スマートスピーカー」内で行っています。その後の指示の部分については、音声データのままインターネット経由でサーバーに送っています。全ての認識を内部で行うためにはかなりの処理能力が必要になり、高価なハードウェアが必要になります。また最初から全てサーバーへ送ってしまうと、あらゆる関係のない音までサーバーで処理することになります。サーバーの処理能力が足りたとしても、今度は「スマートスピーカー」がスパイ行為を行っていると思われてしまう危険があります。実際、偶然最初の呼びかけが入ってしまい、以後家庭内の会話が全てサーバーへ流れていってしまったことがあります。
- インターネットへの接続装置：無線 LAN 経由でインターネットへ接続しますので、「スマートスピーカー」を利用する際には、家庭内に無線 LAN の設備が必要です。音楽を継続的に再生するような場面を考えると、スマホなどの回線を利用するのは費用がかかりすぎるでしょう。

以上の機能をスマートフォンと比べると、スマートフォンから電話機能、タッチパネル付きディスプレイ、メモリー、電池を除いたものとも考えられます。特にタッチパネル付きディスプレイがないために、指示を音声で行い、結果も音声で受け取ることになります。音声での指示は、誤認識されると伝わりません。現在のところ誤認識された場合の対処法が拒否しかないようです。つまり「○○をして」と頼んでも「○○」の部分が認識できなかった場合、「できません」と言う応答になるようです。長い結果を音声で返されると困ります。ちょっと名前を忘れたので「AKB48 のメンバーの名前を教えて」と質問して、48 名分の名前をずらずらと言われても困ると思います。「スマートスピーカー」の設定のようなややこしいものは、大抵スマートフォンの方で行うようになっています。

また利用者が使えるメモリーも最低限しかありません。スマートフォンならその本体に記憶させる個人的な情報も、全てサーバーの方に保存されます。勝手にそれを利用されてもまず分かりません。屋外での利用ができるように、電池を内蔵した「スマートスピーカー」や電源が USB 接続になっているので、モバイル・バッテリー[*26]がつなげるものもあります。

8.2 シナリオの問題

音声で指示を与えるのは手軽ですが、定型的な指示の仕方が確立していないために、実際にスマートスピーカーに指示を与えようとすると、なんと言えば良いのか迷います。実際のお店で買い物の場合では、商品の名前がわからなくても実物があれば、「これください」で買い物ができます。スマホで買い物の場合でも、例えば欲しい服の画像をタップして注文することができます。欲しい服の画像がなかなか出てこないかもしれませんが。音声で買い物でもし商品名がわからない場合はどうすればよいのでしょうか。ここではもう少し具体的に考えてみます。

「Amazon Echo」や「Google Home」などでは応答の仕方を登録することができます。また既に登録された応答を、自分のスマートスピーカーで使えるように設定し利用することができます。ここではその応答の仕方の簡易版モデルを例にします。新幹線のチケット販売サービスとします。行き先は「東京」と「大阪」だけ、

[*25] テレビ番組の「スマートスピーカー」の CM 内の呼びかけに反応することもあるそうな。
[*26] 外出中にスマートホンの電池切れの際に、スマートホンを充電するために使える電池

一度に 5 枚まで買えることにします。何日の何時頃の列車かとか、既にその列車は満席と言うような事はないことにします。表 3 がシナリオの例です。

表 3　チケット販売のシナリオ

話す　「どちらへ行きますか？」
聴く　X
もし　X = 東京
話す　「切符は何枚ですか？」
聴く　Y
もし　Y は 0 より小さいか Y は 5 より大きい
話す　「切符は 1 枚から 5 枚まで買えます」
最初に戻る
そうでなければ
出力　東京　Y 枚
終了
そうでなければ、もし　X = 大阪
話す　「切符は何枚ですか？」

以下省略

大体の動きはわかると思います。スマートスピーカーが聴いた内容が X や Y に入ります。ただその次が問題です。何を言われたのか判断するため、まず「東京」と比較していますがそれで足りるのでしょうか。恐らく大半の利用者は「東京」とだけ答えるでしょうが、「東京です」、「東京だよ」、「えーっと東京」などの答えが来る可能性も十分あります。この辺りならまだ「東京が含まれていたら」と言う条件でなんとかなりそうですが、「え？ごめん。なんて言ったの？」と来られると困ります。

次に Y に枚数に関する聴いた内容が入りますが、同じ数でも「さん」、「さんまい」、「みっつ」など色々ありそうです。この点同じように音声でやり取りをする電話での自動応答は、数字入力を電話の数字のボタンが利用できるので有利です。電話機の数字といくつかの記号のボタンを押すと、ボタンごとに異なる組み合わせの音がするので、その音で押されたボタンを確実に認識することができます。よって質問もできるだけ答えが数字になるようにしています。例えば「東京なら 1 のボタンを、大阪ならば 2 のボタンを押してください」のようにします。選択肢が多くなると何番だったかわからなくなるので、通常途中で答えのボタンを押しても良いようになっています。ボタンの音と音声が混ざっても、異なる高さの音の組み合わせで認識を行っているので誤認識はあまりないようです。

数字を誤認識したり、枚数がサービス範囲を超えていた場合、表 3 では最初に戻ってしまいますが、もう少しマシな対応にしないと利用者を怒らせることになるでしょう。また「出力　東京　Y 枚」は別のシステムへデータを送ることを想定していますが、黙ったまま「終了」していますので、利用者はしばらく「どうしたの？」と待つことになるでしょう。利用者に対して常に適切な応答をしないと同様の事態を招くことになりますし、しつこく応答するとうざいシステムになってしまいます。

9. AI に対抗するには

本来は「深層学習」の章に入れる内容ですが、この内容を入れることを思いついたのが 2020 年になってからだったので、最後のおまけのように付けました。内容は国立情報学研究所教授の新井紀子著「AI vs. 教科書が読めない子どもたち」(2018 年、東洋経済新報社) の紹介です。

新井氏は 2011 年から「ロボットは東大に入れるか」(東ロボ) と言う人口知能プロジェクトを開始しました。このプロジェクトは東京大学の入試問題を人工知能に解かせることを目標としたものです。十分な研究費がなく、大学院生も含めると 100 名近い研究者のボランティア活動によって進められたこのプロジェクトは、2013 年のセンター模試の成績は偏差値 45 程度でしたが、2016 年のセンター模試では偏差値が 57.1 まで上がり、センター試験のみの入試であれば、全国に 127 校ある国公立大学のうち、23 大学の 30 学部の 53 学科で合格率 80% 以上という判定が出ました。さらに数学と世界史の 2 教科のみは「東大入試実戦模試」(筆記試験) で偏差値 61.8 と 76.2 と言う好成績をあげています。ただこのプロジェクトは終了しました。それはこのプロジェクトによって、いくつかの大きな壁が明らかになり、それを越える方法が出てくるまで先へ進めないことがわかったからです。量子コンピュータやビッグデータでも越えられなない壁です。新井氏はこれから「AI が神になる？・・・なりません！」、「AI が人類を滅ぼす？・・・滅ぼしません！」、「シンギュラリティが到来する？・・・到来しません！」と述べています。ただ囲碁や将棋で AI にプロが勝てなくなったのと同様に、分野を絞ってセンター模試であれば、高校生の 8 割近くは AI に負ける事が明らかになりました。

これからの世の中ではどんどん AI が使われるようになって、その分の雇用が失われるでしょう。一説によればそれは 50% 程度にもなります。最近の別の研究では 10% 程度とも言われますが、それでも国内だけで数百万人が職を失います。その一方で新しい雇用が生じるのですが、AI のために失業した人が雇用されるわけではありません。かつて産業革命が起こったときにも同様の事が生じましたが、現在その当時の産業革命による失業は残っていません。これは産業革命後の社会に適応した人々が段々生まれて、新しい雇用の対象になったからです。同様に AI による失業も時間が経てば解消される事が予想されますが、それには時間がかかるので現在や近い将来失業する人は救われません。

とりあえず AI によって奪われるような仕事を避けるのが良いでしょうが、それはどのような仕事か。これについては「コミュニケーション能力や理解力が求められる仕事」や「柔軟な判断力が必要な肉体労働」ではないかと言われています。必要なスキルは「高度な読解力と常識、加えて人間らしい柔軟な判断能力」となります。新井氏はこの中から、常識はみんな持っているだろう、としています。みんなが知らない知識は常識にならないからです。柔軟な判断もできるだろうとして、問題は「読解力を基盤とする、コミュニケーション能力や理解力」の有無です。そして新井氏が日本の中高校生を対象に行った調査の結果、残念ながら多くの中高生は教科書の記述を正確に読み取ることができません。そして読解力は高校以下で獲得するのが普通で、それ以降はあまり伸ばす機会がないので多くの大人も同様のレベルと考えられます。

読解力の調査では東ロボプロジェクトで得られた知見が使われています。問題の文章を AI で理解させようとするとき、AI でも割とできるものや、AI ではなかなかできないものがあります。主語と述語、修飾語と被修飾語のペアを見つける「係り受け」解析や「それ」「これ」と言った指示代名詞が何を示しているのか求める「照応」解析は東ロボでも 80% ぐらいまでできました。一方 2 つの異なる文が同じ意味であるかどうか判定する「同義文判定」は長年研究されていますがうまく行きません。AI がなかなか身に付けられない常識が必要なものとして「推論」、「イメージ同定」、「具体例同定」も取り上げています。「推論」は文章に書かれていない常識も用いた意味の理解です。「イメージ同定」は文章と図やグラフを比較して同じ内容のもの見つけます。「具体例同定」は文章にかかれている内容の具体例となるものを見つけます。これらは意味を理解できない AI が苦手とするものです。元となる文章は教科書と新聞記事を利用しています。あまり専門的な文章では難しい用語が理解の妨げになる恐れがあるからです。また新井氏は教科書の意味が分からないのでは授業にならないので、これぐらいはできるだろうと思っていたようです。

調査した人数は大人も含めて 2018 年までの段階で 2 万人です。その結果は新井氏の本に表やグラフと共に詳しく書かれています。ここでは、その中の表を一つ紹介します。

表 4　学年別のランダム率 (%)

学年	係り受け	照応	同義文判定	推論	イメージ同定	具体例同定 (辞書)	具体例同定 (数学)
小 6	35.8	52.6	95.3	57.3	54.7	56.0	100
中 1	31.6	36.8	78.9	61.8	42.4	68.5	86.4
中 2	26.1	25.4	78.8	54.6	37.3	63.5	82.4
中 3	18.3	15.6	70.2	43.4	31.1	48.7	79.4
高 1	10.8	6.5	57.4	38.6	11.7	47.9	53.1
高 2	10.3	6.2	65.7	37.6	13.4	53.4	57.6

この表 4 は調査の結果から回答がランダム、つまりサイコロ振って答えているのと変わらない回答の割合の表です。よって数字が小さいほど理解して答えている人が多いことになります。残念ながら「同義文判定」や「具体例同定」は高校 2 年生でも半分以上の人が間違えています。

外国の調査結果に比べるとこれでもましだそうです。ただ外国は母国語が違う人の割合が高い国が多いので、読解力のテストでは不利な状況にあります。母国語が日本語の人が大多数の日本ならば、本来もっと高い読解力があるはずです。読解力を伸ばす簡単な方法は見つかっていません。新井氏は統計的に様々な要因との関連を調べたそうですが、家庭の貧困が悪い影響を与えているぐらいしか見つからなかったそうです。この調査に協力的だった埼玉県の戸田市の小中学校は、読解力の残念な結果を受けて、様々な読解力の向上を試みました。するとこれまで埼玉県で中位レベルの学力だった戸田市の小中学校が、埼玉県で学力が総合 1 位レベルになったそうです。授業が分からないのは内容が難しいからでなく、単に読解力が足らないためかもしれません。そして社会に出てからも、仕事をするために様々な勉強が必要になります。教科書が読めなければ AI がやって来るまでもなく、仕事ができないことになります。