パラメトリックスピーカによる防災誘導の向上

大木, 成文 / OHKI, Sigefumi

(発行年 / Year)
2009-03-24
(学位授与年月日 / Date of Granted)
2009-03-24
(学位名 / Degree Name)
修士(工学)
(学位授与機関 / Degree Grantor)
法政大学 (Hosei University)

２００８度　修士論文

パラメトリックスピーカによる防災誘導の向上

Effect of Evacuation Announcement by Using
Parametric Loudspeakers

指導教授　　小林　尚登

大学院工学研究科
電気工学専攻修士課程
07r3108
オオキ　シゲフミ
大木　成文

Abstract

This research is on an application of parametric loudspeakers. Parametric loudspeakers have sharp directivity, and the sound reaches farther distance than ordinary speakers. Then, we propose to use these parametric loudspeakers in case of urban disaster. By the effect of sharp directivity, the magnitude of the reflected sound among buildings decreases sharply.

Thus, the parametric loudspeakers provide clear announcement to the people on the streets. This paper shows the effect of evacuation announcement by using parametric loudspeakers.

本論文の構成

本論文は，全体が５章から構成される．

１章では，本研究の背景について述べる．第２章ではパラメトリックスピーカ及び従来型スピーカの音響特性を，数値解析により比較する．そして，第３章では実機による検証，第４章ではシミュレーションによる検証を行う．そして最後に，第５章で結論を述べる．

目 次

図 目 次

第1章

序論

近年，多発する災害に対し，防災対策への関心が高まっている．特に都市災害に対しては，建物や人口の密集化により，被害が大規模になる恐れがあり，被害を最小に抑える対策が必要である．防災対策の試みとしては，音響放送による避難誘導が一般的である．しかし音響放送には，建物の密集した場所では，音の反響により情報が聞き取り難いという問題点がある．

そこで，我々はパラメトリックスピーカに注目した．パラメトリックスピーカとは，超音波を用いることにより，指向性が高く，特定領域でのみ音を聞くことができるスピーカである．通常型スピーカとパラメトリックスピーカの音の届き方の比較例を Fig.1.1 に示す．この技術を用いれば，周りに音が漏れない為，情報を必要としている者だけに提供することができる．

つまり，一般的なスピーカでは音に広がりがある為，ビル等の建設物に音が反射し、反射音が生じる．そして，直接音と反射音とで、到達時間の差が生じ，情報の聞き取りが困難になると考えられる．しかし，パラメトリックスピーカの場合，指向性が高く，音の広がりが少ない為，到達時間の差が軽減される．これにより，明瞭な音声案内が実現でき，人員的被害の拡大を軽減できると考えられる．

本研究では，都市災害において明瞭な音声案内を提供するシステムの提案・開発を行い，人員的被害拡大の軽減を目的とする．特に本稿では，パラメトリックスピーカの作製・実験，及び実際の災害を想定したシミュレーションを行った．

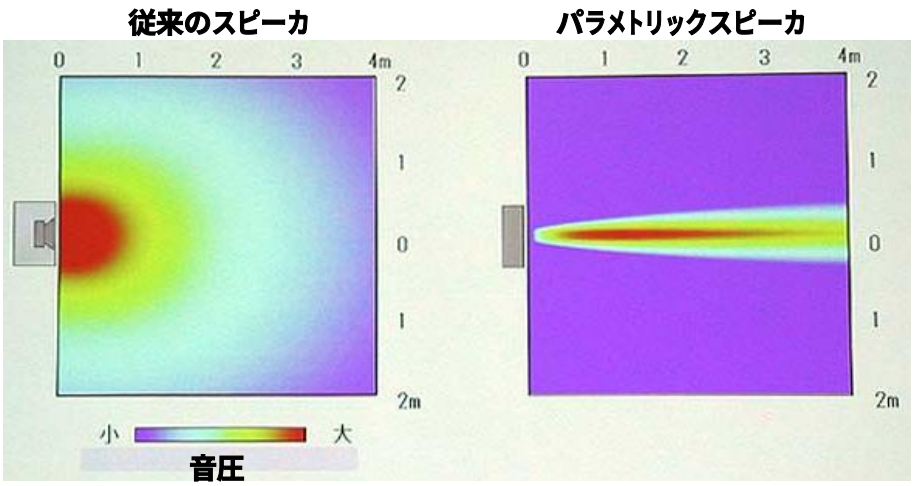

Fig.1.1　音の届き方の比較

第2章

音響特性

通常型スピーカ及びパラメトリックスピーカの音圧及び指向性に関する音響特性の比較を行った.

通常型スピーカの指向性パターン及び音圧特性は，半径aの円板が正弦ピストン振動していると仮定すると，以下のように記述できる．[1]

$$D_{(\theta)} = \frac{2J_1(ka\sin\theta)}{ka\sin\theta} \tag{1}$$

$$P_{(r)} = j\omega\rho_0 \frac{2\pi a^2 V_0}{4\pi r} e^{-jkr} D_{(\theta)} \tag{2}$$

ここで，J_1は第一次ベッセル関数，V_0は音源の振動速度である.

一方，パラメトリックスピーカの指向性パターン及び音圧特性は，以下のように記述できる．[2]

$$D_{(\theta)} = \frac{1}{\left\{A^2 + \left[2K\sin^2(\theta/2)\right]^2\right\}^{\frac{1}{2}}} \tag{3}$$

$$P_{(r)} = \frac{p_{01}p_{02}\Omega^2 S}{4\pi\rho_0 {c_0}^4 r}\exp(-\alpha\_r)D_{(\theta)} \tag{4}$$

ここで，$A=\alpha_1+\alpha_2-\alpha_-$，$\alpha$ は吸音係数，$K=k_1-k_2$，k は波数，r は音源からの距離，p_{01} と p_{02} は音源の音圧，$\Omega=\omega_1-\omega_2$，ω_1 と ω_2 は搬送波の角周波数，S はビーム幅，ρ_0 は空気の密度，c_0 は音速である．

以上(1)～(4)式により得られた数値解析結果を Fig.2.1，Fig.2.2 に示す．このとき，各パラメータは Table.2.1 のように設定した．また，Fig.2.1 では，スピーカからの正面距離 100[m]での音圧を表す．

Fig.2.1 より，パラメトリックスピーカの指向性の高さが確認できた．また，Fig.2.2 より，パラメトリックスピーカの減衰率の低さが確認できた．

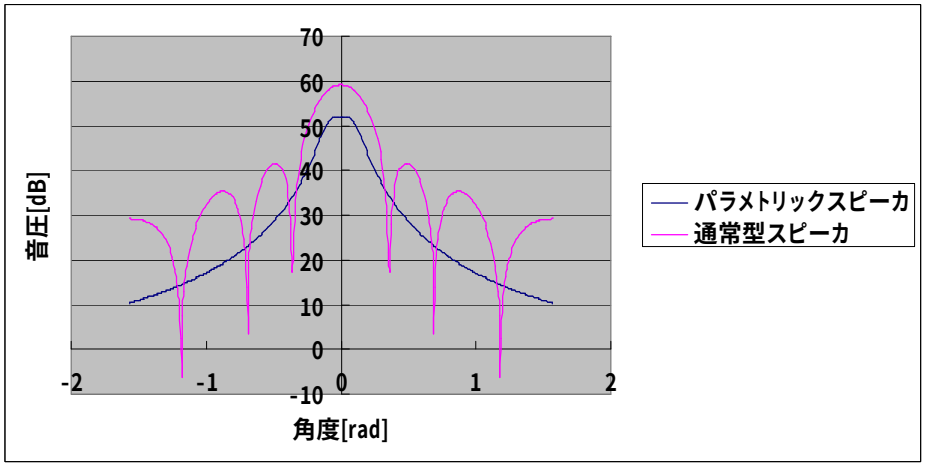

Fig.2.1　指向性に関する比較

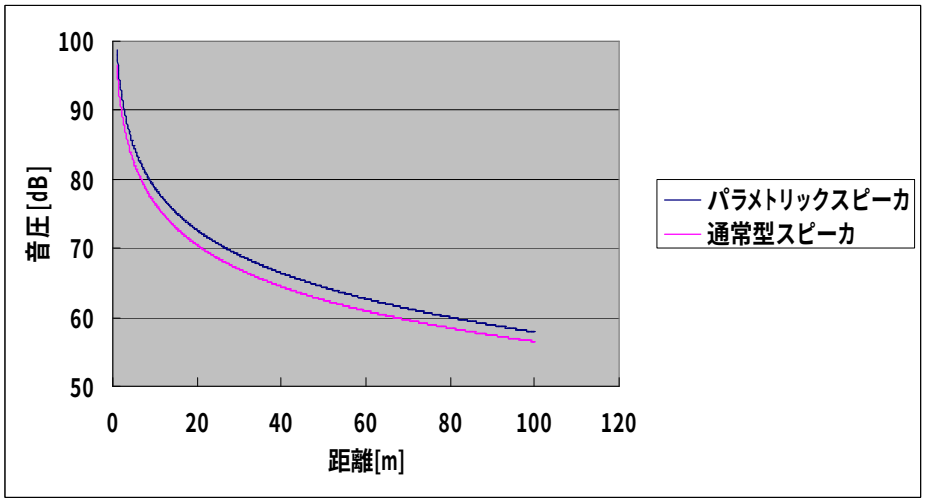

Fig.2.2　減衰率に関する比較

Table.2.1　特性式のパラメータ

A	0.31
K	$36.7[m^{-1}]$
p_{01}, p_{02}	$50[Pa]$
Ω	$1.26\times10^{4}[rad]$
S	$0.28[m^{2}]$
ρ_0	$1.18[kg/m^{3}]$
c_0	$343[m/s]$
a	$0.3[m]$
V_0	$2.0\times10^{-3}[m/s]$

第3章

実機による検証

パラメトリックスピーカは，空気の非線形性により生じる仮想的な音源効果を利用している．二つの周波数の異なる超音波を同時に放射し，重ね合わせると，伝搬過程で波形が歪み，差周波数音が可聴音となり得られる．例えば，40[kHz]と 42[kHz]の二つの周波数の超音波を同時に放射すると，差周波数である 2[kHz]の音が可聴音となり生成される．同時に，結合波や高調波も発生するが，超音波帯域の為，差周波数のみ可聴音として聞こえる．パラメトリックスピーカでは，超音波をオーディオ等の信号波で振幅変調し空気中に放射することにより，この現象を実現できる．

3.1 振幅変調

振幅変調とは，ある情報（信号波）を特定の基準周波数（搬送波）の振幅の強弱で伝搬する変調方式である．振幅の強弱は，信号波と搬送波の積により得られる．基準周波数を $A\cos 2\pi f_c t$ ，信号波を $B\sin 2\pi f_s t$ とすると，変調波は

$$v_{am}(t) = A\cos 2\pi f_c t \cdot B\sin 2\pi f_s t$$

$$= \frac{AB}{2}(\sin 2\pi(f_c + f_s)t + \sin 2\pi(f_c - f_s)t) \qquad (5)$$

と表せる．

(5)式の第一項目から，搬送波と信号波の周波数の和成分が生成されていることが確認できる．

パラメトリックスピーカで任意の情報を再生する場合，二つの超音波の差周波数の音が可聴音となり生成される為，基準周波数を 40[kHz]，情報となる信号波を a とすると，(40+a)[kHz]の周波数を生成する必要がある．つまり，搬送波と信号波の周波数の和成分を生成する為に，搬送波と信号波の掛算を行う．

3.2 機構構成

パラメトリックスピーカは，変調回路，スピーカ部，音声入力部から構成される．音声入力部から得た音声信号を周波数 40[kHz]の搬送波と共に変調回路に入力し，変調を行う．そして，変調回路より出力された変調波をスピーカ部に入力し，空気中に放射する．

3.2.1 変調回路

変調回路には，アナログ掛算器 NJM4200 を使用する．Fig.3.1 の回路により，二つの入力の掛算が可能となり，出力V_0は，

$$V_0 = \frac{R_0 R_2}{R_1^{\,2}} \frac{V_X V_Y}{V_R} \tag{6}$$

で表せる．ここで，V_Xは信号波，V_Yは搬送波（40kHz），基準電圧V_Rを15[V]とし，

$$V_0 = \frac{1}{5} V_X V_Y \tag{7}$$

となるように，パラメータを決定した．

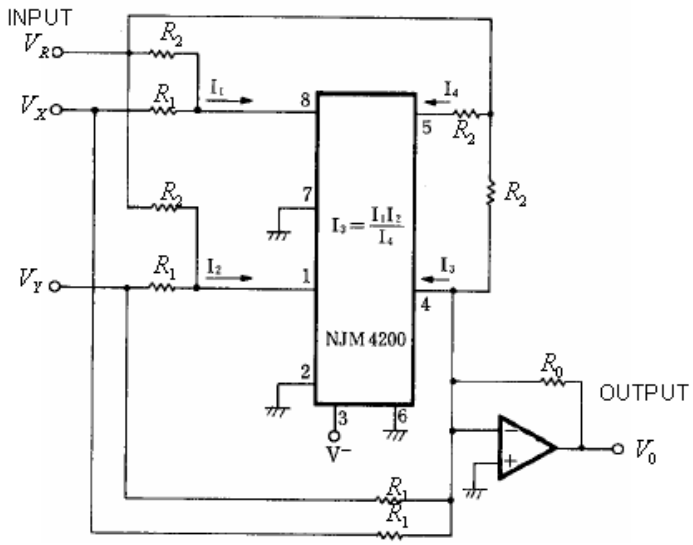

Fig.3.1　NJM4200 による掛算回路

3.2.2　スピーカ部

スピーカ部は，スーパツィータから構成される．スーパツィータにはフォスター電機㈱製 FT17H を使用する．スーパツィータ FT17H は，再生周波数帯域 5[kHz]～50[kHz]，出力音圧レベル 98.5[dB/W(1m)]である．

また，より音の広がりを防ぐ為，スーパツィータの先端部に，吸音材で作製した筒状のホーンを接続した．吸音材には，東京防音㈱製ホワイトキューオンを使用し，厚さ 5[cm]，密度30[kg/m^3]のものを採用する．ホワイトキューオンの周波数特性を Fig.3.2 に示す．

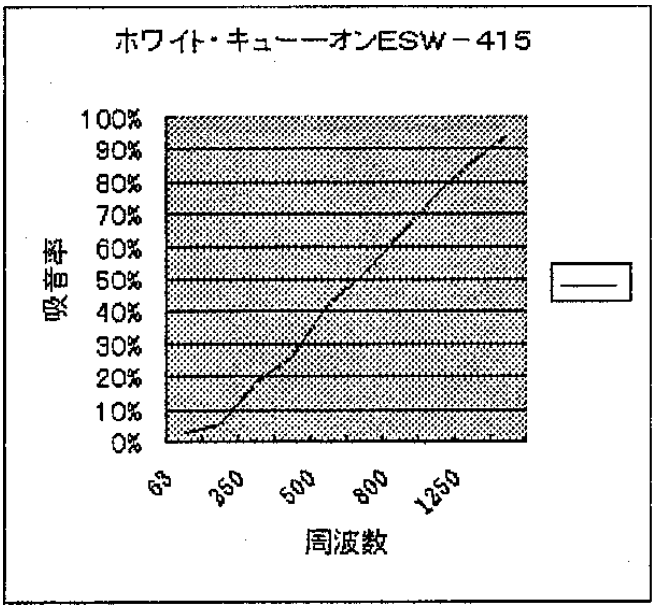

Fig.3.2　ホワイトキューオンの周波数特性

3.2.3　音声入力部

音声入力には，オーディオケーブルを用い，PC からの音源を回路に入力する．これにより，音声のみだけでなく，音楽等のオーディオファイルの再生も可能となる．

3.3 実験

試作したパラメトリックスピーカにより，以下の二つの実験を行った．本研究では，情報の聞き取りに関する研究である為，人間の耳による測定を行った．

3.3.1 可聴角度の測定

[1] 実験方法

人間の耳により，スピーカからの正面距離での最大可聴距離及び平行方向での最大可聴距離を測定し，試作したパラメトリックスピーカの可聴角度を求める．測定方法のイメージ図を Fig.3.3 に示す．このとき，再生周波数 2[kHz]で正面距離 2[m]の位置での音圧が 70[dB]になるように設定した．

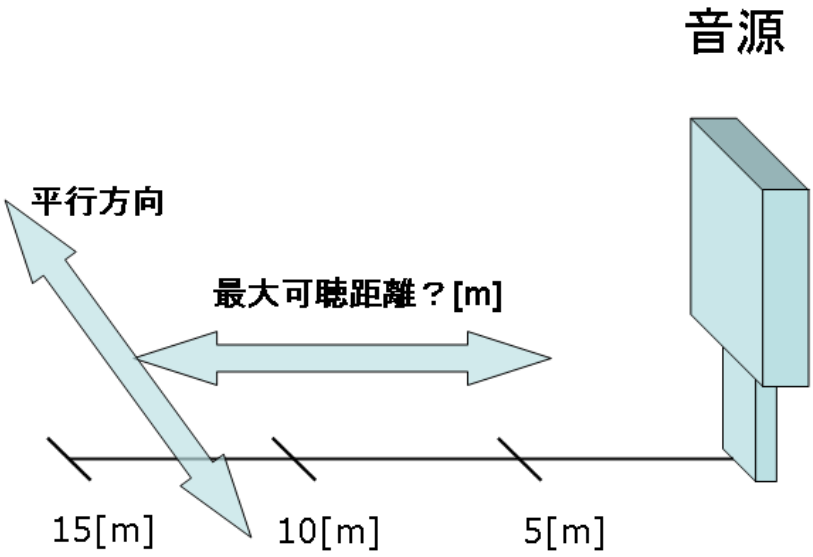

Fig.3.3 可聴角度の測定方法

[2] 実験結果

測定の結果，直線距離での最大可聴距離は約 18[m]となった．また，平行方向での可聴距離は左方向，右方向共に約 6[m]となり，パラメトリックスピーカの可聴角度は±18.4°となった．

3.3.2 認識率についての検証

[1] 実験方法

スピーカの両側に音の反響しやすい壁がある場合において，通常型スピーカとパラメトリックスピーカとの音の聞こえ方の違いを確認する．

また，パラメトリックスピーカからの正面距離 2[m]での音圧を 75[dB], 85[dB], 100[dB]と設定したときの，最大聞き取り可能距離を測定する．このとき，スピーカからランダムな単語を放送し，聞き取り確認を行うことで，情報の聞き取り可能な最大距離を測定する．

さらに，最大聞き取り可能距離で，通常型スピーカとパラメトリックスピーカとで，それぞれ 10 回の単語聞き取り確認を行い，何回聞き取れたかを調査する．

[2] 実験結果

実験結果を Table.3.1 に示す．実験の結果，スピーカの両側に音の反響しやすい壁がある場合，通常型のスピーカでは音に多少の歪みをあるのに対し，パラメトリックスピーカでは音に明瞭感があることが確認できた．この歪みは，音源からの距離が遠くなる程，顕著に現れた．表２からも，パラメトリックスピーカの方が通常型のスピーカよりも情報の認識確率が高いことが確認できる．これは，パラメトリックスピーカの指向性の高さにより，音の反響が軽減された為だと考えられる．このことから，パラメトリックスピーカの方が，通常型のスピーカよりも遠距離での情報の聞き取りが可能であるといえる．

Table.3.1 音圧に対する認識確率の比較

	観測条件(設定音圧:観測距離)		
	75[dB]:25[m]	85[dB]:38[m]	100[dB]:45[m]
通常型スピーカの認識確率	2/10	4/10	4/10
パラメトリックスピーカの認識確率	8/10	8/10	10/10

第4章

シミュレーションによる検証

大規模災害における防災の分野では，現実に実験を行うことが困難な点から，シミュレーションによる避難行動や誘導方法の分析が広く行われている．従来の研究としては，玉川ら[3]による津波災害を想定した避難シミュレーションや，岡田ら[4]による誘導者を用いた避難誘導シミュレーションが挙げられる．しかし，災害時に避難を促す際，音響放送による誘導方法を用いることにより，避難者全体を迅速に誘導できると考えられるが，音による効果を示した研究はなされていない．さらに，パラメトリックスピーカを用いることで，音響放送の問題点である，音の反響による聞き取り難さの軽減が期待できる．

そこで，本章ではパラメトリックスピーカを大規模災害に使用した場合の効果をシミュレーションにより検証する．特に，本研究では都市部における災害を想定している為，東京都庁周辺を対象地域とし，シミュレーションを行う．

4.1 反響による影響

道幅を x，音源からの距離を r とし，距離 r[m]先まで建物に 1 回も反射しない場合の可聴角度を

$$\theta = \tan^{-1}\frac{x/2}{r} \tag{8}$$

と仮定する．都庁周辺では道幅が約 20[m]である為，スピーカからの距離 200[m]先まで 1 回も音が反射しない角度は，(8)式より±2.86°となる．試作したパラメトリックスピーカの可聴角度は，3.3.1 の実験から，±18.4°であることが確認できた．この場合，都庁周辺では，200[m]先までの音の反射回数は 3 回となる．一方，通常型のスピーカの可聴角度を±45°とすると，200[m]先までの音の反射回数は 20 回となる．

そこで，建物との反射回数が 1 回，3 回及び 20 回における音の反射による影響を，音の到達時間及び音の減衰に関して検証する．

4.1.1　音の到達時間

直接音と反射音との到達時間の差を以下の方法で計算し，シミュレーションを行った.

音源から観測者までの距離を x，音速を c とすると，観測者までの到達時間は,

$$t_1 = x / c \tag{9}$$

となる.

また，道幅を r としたとき，n 回反射したときの観測者までの到達時間は,

$$t_2 = \frac{x}{\cos(\tan^{-1}(nr / x)) \cdot c} \tag{10}$$

となる.

よって，直接音と反射音の到達時間の差は,

$$T = t_2 - t_1 \tag{11}$$

で求まる.

4.1.2 音の減衰

距離に対する音の減衰についても，以下の方法でシミュレーションを行った．距離xに対する音圧は，道幅を r [m]，反射回数を n 回，壁の吸音率をαとすると，(2)式より，

$$P_{n(X)} = (1-\alpha)^n \, j\omega\rho_0 \frac{2\pi a^2 V_0}{4\pi X} e^{-jkX} \tag{12}$$

$$X = \frac{x}{\cos(\tan^{-1}(nr/x))} \tag{13}$$

と表せる．

4.1.3 結果と考察

(9)〜(11)式より求めた, 観測距離に対する到達時間差の特性を Fig.4.1 に示す．ここで，道幅 r を 20[m]，音速 c を 343[m/s]とし，反射回数 1 回，3 回及び 20 回とした.

また，(12)，(13)式より求めた，反射回数 1 回，3 回及び 20 回とした場合の，観測距離に対する減衰率の特性を Fig.4.2 に示す．ここで，道幅 r を 20[m]，吸音率αを 0.02 とした.

Fig.4.1 から，反射回数が多くなるに従い，直接音との到達時間の差が徐々に大きくなることがわかった．また，Fig.4.2 から，反射回数が多くなるに従い，音圧が低くなることが確認できる．さらに，観測距離が遠くなる程，直接音と反射音の音圧が近くなることがわかった.

つまり，反射回数が多くなる程，微妙に時間差のある音が多くなり，その時間差も大きくなる．さらに，スピーカからの距離が遠い場所では，時間差のある音との音圧が近くなる．この結果から，ビル等の建物が密集した空間では，反射回数が多いほど，情報供給が困難になると考えられる.

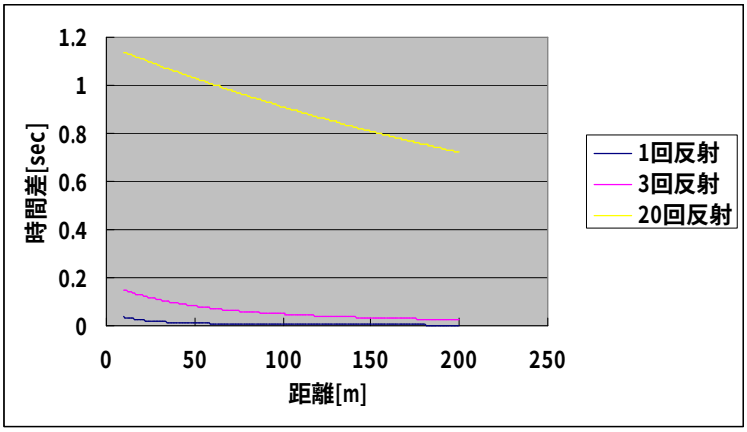

Fig.4.1　直接音と反射音の到達時間差

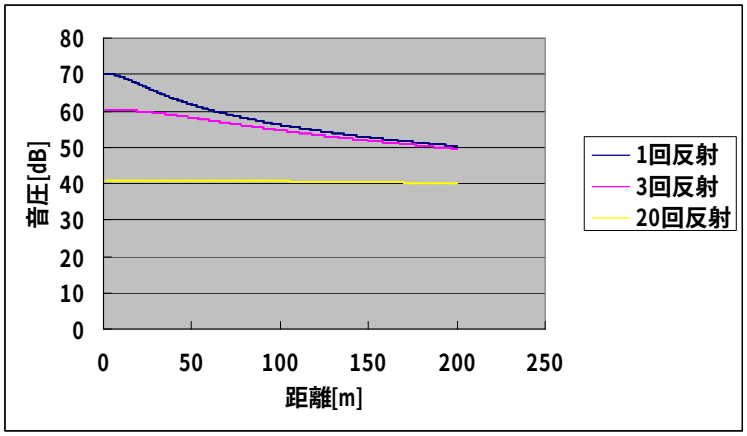

Fig.4.2　直接音と反射音の減衰率

4.2 避難行動に対する影響

パラメトリックスピーカを防災放送として使用した場合の避難行動への影響を, マルチエージェントシステムを用い, シミュレーションを行う. マルチエージェントシステムとは, 個々のエージェントにルールを与え, 複数のエージェントの動きから全体の現象を捉えようとする手法である.

ここでは, (株)構造計画研究所のマルチエージェントシミュレータ「KK-MAS」を用い, 避難シミュレーションを行う.

4.2.1 空間のモデル化

本モデルでは, 都庁周辺地域の二次元空間上, 約 700[m]×700[m]に 100×100 マスの空間格子をとる.

避難行動モデルの構成要素は, 避難者, 障害物, 出口から構成される. 障害物は建物, 出口は避難場所の入り口を表す. 避難者はシミュレーション開始時に障害物, 出口に重ならないようにランダムに配置され, 障害物に重ならないように, 出口を目指し移動する. そして, 全避難者の避難完了後, 所要ステップ数を表示する. このとき, 避難者の移動速度は参考文献[5]を考慮し, 1[m/s]～2[m/s]の範囲で各エージェントに変化を与えた. 本モデルでは, 避難者は 1 ステップで 1 セル移動する. 1 セルが約 7[m], 避難者の平均移動速度が 1.5[m/s]である為, 1 ステップ 4.67 秒と設定する. また, 避難者の人数はコントロールパネルにより, 10 人～1000 人の範囲で設定できるようにした.

以上のように, 空間をモデル化し, 通常型スピーカを使用した場合, パラメトリックスピーカを使用した場合, それぞれの避難者の移動を比較する.

4.2.2　誘導方法の違い

災害時の避難場所は，集団での避難に対応する為，およそ 10 ヘクタール以上が必要だとされている．そこで，都庁周辺で災害が発生した場合の避難場所を，新宿中央公園とする．

防災放送のスピーカの設置場所を，通常型スピーカを使用した場合，パラメトリックスピーカを使用した場合，それぞれ Fig.4.3，Fig.4.4 に示す．Fig.4.3 では，水色の丸印がスピーカを，橙色の丸印が音達距離を表す．Fig.4.4 では，水色の丸印がスピーカを，赤色の矢印が音達距離を表す．

従来の防災放送の音達距離は約 300[m]である．また，設置場所は主に役所，学校及び公園である．そこで，通常型スピーカを使用した場合のスピーカの設置場所を，新宿中公園とした．一方，パラメトリックスピーカを災害誘導に利用した場合，指向性が高く特定領域に音声案内ができる為，Fig.4.4 のようにスピーカを設置することで，道に応じた誘導案内が期待できる．例えば，現在いる場所から避難場所までの移動方向を道ごとに放送することが可能である．

これにより，従来の防災放送では，災害情報及び避難場所の案内のみだが，パラメトリックスピーカを使用することにより，道に応じた避難場所への移動方法の案内が可能となり，災害による被害の拡大を防止できると考えられる．

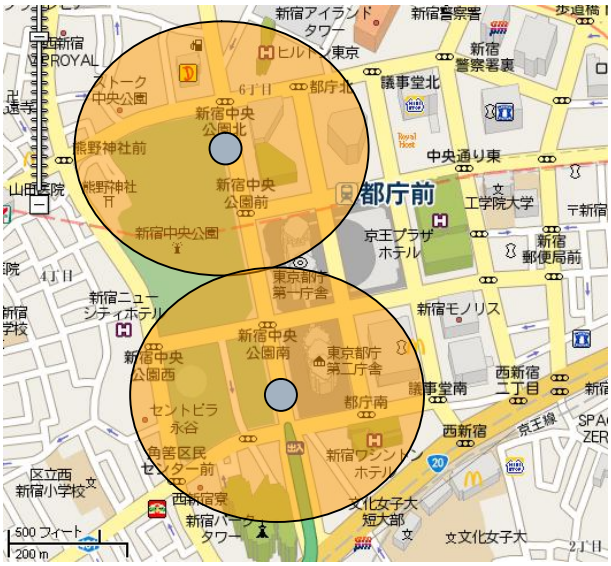

Fig.4.3　通常型スピーカの設置場所

Fig.4.4　パラメトリックスピーカの設置場所

4.2.3 エージェントルール

通常の避難において，避難場所の情報が聞こえても，避難場所への道順を知っている人と知らない人では，避難時間に差が生じる．そこで，通常型のスピーカでの防災放送では，音声案内が聞こえるか，避難場所までの道順を知っているかの要素について，エージェントの行動を差別化する．

音声案内が聞こえるかどうかは，Fig.4.5 に示すように，スピーカから半径 200[m]以内に存在するエージェントを音声案内が聞こえるとする．それ以外は，音声が聞こえても建物との反響により内容が聞き取れないと仮定した．また，避難場所までの道順を知っているエージェントの割合は，あらかじめ全体人数の 1 割ごとに変化できるように設定した．

道順を知っているエージェントの行動は，音声案内の聞こえるエリアにいた場合，エージェントは避難場所までの最短経路を求め，移動する．また，音声案内の聞こえないエリアにいた場合には，道なりに沿ってランダムに移動するように設定した．

道順を知らないエージェントの行動は，まず周囲一回り分のセルを見渡す．このとき，周りに道順を知っている且つ音声が聞き取れているエージェントが存在する場合，道順を知らないエージェントは道順を知っているエージェントと同じ方法で移動する．それ以外は，たとえ音声案内が聞こえていても，避難所までの道順がわからない為，道なりに沿ってランダムに移動する．

一方，パラメトリックスピーカで音声案内を行った場合のエージェントの誘導には，道に応じた音声案内が行える．このため，避難場所までの道順を知っているかどうかは関係なく，エージェントが存在する場所に応じて Fig.4.6 に示す通り，移動するように設定した．

Fig.4.5　通常型スピーカを使用時の避難可能エリア

Fig.4.6　パラメトリックスピーカ使用時の移動パターン

4.2.4 結果と考察

避難者の数を 200 人，通常型スピーカを使用した場合の避難場所までの道順を知っている人の割合を 4 割として，シミュレーションを行った．シミュレーションの実行画面，通常型スピーカを使用した場合の残留歩行者数の時間推移，パラメトリックスピーカを使用した場合の残留歩行者数の時間的推移を，それぞれ Fig.4.7，Fig.4.8，Fig.4.9 に示す．また，通常型スピーカを使用した場合，パラメトリックスピーカを使用した場合，それぞれの避難誘導シミュレーションを 10 回ずつ行い，避難完了ステップ数の平均値を求めた．シミュレーションの結果，通常型スピーカを使用した場合，パラメトリックスピーカを使用した場合，それぞれの平均ステップ数は 222.8，102.9 となった．本モデルでは，1 ステップ 4.67 秒と設定している為，実時間での避難完了時間は，通常型スピーカを使用した場合で 17.3 分，パラメトリックスピーカを使用した場合で 8.0 分となる．よって，避難完了時間の差は，パラメトリックスピーカを使用した方が，約 9.3 分早く避難できる結果となった．

この結果，災害時の誘導案内にパラメトリックスピーカを使用し，避難者個人が避難場所までの道順を知ることにより，通常型スピーカを使用した場合よりも，約 2 倍早く避難できることが確認できた．

Fig.4.7　シミュレーション実行画面

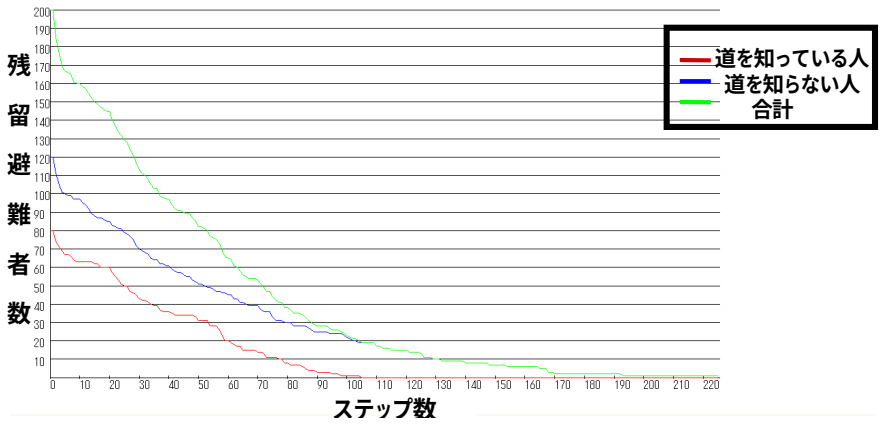

Fig.4.8　残留歩行者数の時間推移(通常型スピーカ)

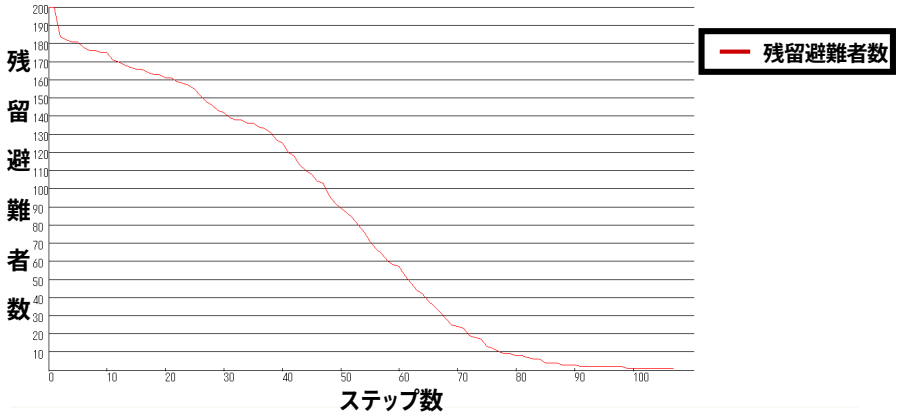

Fig.4.9　残留歩行者数の時間推移(パラメトリックスピーカ)

第5章

結論

本稿では，パラメトリックスピーカを使用することにより，都市災害において，明瞭な音声案内を提供するシステムの提案を行い，試作機による検証及び実際の災害を想定した避難誘導シミュレーションを行った．今回提案したシステムにより，明瞭な音声案内の提供及び避難時間の短縮を示すことができた．今後の課題としては，避難誘導シミュレーションの精度向上による，実際の災害状況に即した検証が挙げられる．

本研究では，パラメトリックスピーカの応用例として，受け手側に有益な効果を与えた．一方，別の利用方法として，受け手側に害を及ぼす使用方法もある．例えば，農作物等に被害をもたらす鳥を追い払う際，パラメトリックスピーカを使用することにより，近隣住民への騒音を立てずに済む．このように，パラメトリックスピーカは様々な場面での応用が期待できる．

謝辞

本研究における知識やアドバイス等を与えて下さり，そして様々な面でご指導いただきました小林尚登教授に感謝の意を申し上げます．

そして知識の乏しい私を様々な面でサポートして下さった小林研究室の先輩方，同輩には大変お世話になりました．特に，川端先生には正規の勤務地における活動がお忙しいにもかかわらず，わざわざ法政大学まで足を運んでいたき，豊富な知識と経験でご指導アドバイスをいただきました．

最後に，6年間大学生活を様々な面で援助して下さった両親に対し，感謝の意を申し上げて本論文の結びとさせて頂きます．

2009年3月　　大木　成文

参考文献

[1] 城戸健一：音響工学，コロナ社，1982

[2] H. O. Berktay：J. Sound Vib.，2(4)，pp435-461，1965

[3] 玉川，大畑，高井，鏡味：日本建築学会北海道支部研究報告集 No.79，2006

[4] 岡田，竹内：避難時における指差し誘導法及び吸着誘導法に対するシミュレーション，法政大学情報メディア研究センター，2007

[5] 円谷，水野，大宮，森田，若松：被験者実験による避難者モデルの定式化－ポテンシャル法に基づいた避難シミュレーションの開発（その１）－，日本建築学会大会(関東)学術講演梗概集，2006

[6] 西巻正郎：電気音響振動学，コロナ社，1978

[7] H.F.Olson：音響工学[上巻]，近代科学社，1959

[8] 鳥飼安生：円形ピストン音源付近の音場－(第２報)音場の理論－，日本音響学会誌、第 14 巻第 1 号、1958

[9] 鎌倉友男：非線形音響学の基礎，愛智出版，1996

[10] 青木，鎌倉，熊本：パラメトリックスピーカ－音場特性と最適変調方式－，電子情報通信学会論文誌，A J74-A，pp332-337，1991

[11] 鎌倉，青木，酒井，銭:パラメトリックスピーカに関する最近の研究，電子情報通信学会信学技法，US2004-94，2005

付録　A

シミュレーションプログラム

提案した避難シミュレータは，空間全体のルールであるワールドルール，及び避難者自体の動きとなるエージェントルールにより構成されている．そこで，通常型スピーカを使用した場合，パラメトリックスピーカを使用した場合，それぞれのワールドプログラム及びエージェントプログラムを示す．

A.1　通常型スピーカを使用した場合の避難誘導

A.1.1　ワールドプログラム

```
Agt_Init{
        Dim i as integer
        dim ninzu as double
        dim ratio as double
        dim siteru as double

        ninzu = world.人の数
        ratio = world.知ってる率
        siteru = (ninzu * ratio / 10) -1
        For i = 0 to siteru
                _CreateAgent(WORLD.街.歩行者)
                //world.街.歩行者.知ってる(i) =true
                world.街.歩行者.色(i) = COLOR_RED
                //_DebugStr(1)
        Next i

        For i = 0 to (ninzu - siteru) -2
                _CreateAgent(WORLD.街.歩行者２)
                //world.街.歩行者.知ってる(i) =false
                world.街.歩行者２.色(i) = COLOR_BLUE
                //_DebugStr(2)

        Next i

        //_DebugStr(siteru)
        _RandomPutAgent(WORLD.街.歩行者, false)
        _RandomPutAgent(WORLD.街.歩行者２, false)
```

```
        World.残留歩行者数 = _CountAgent(World.街.歩行者)
        World.残留歩行者数２ = _CountAgent(World.街.歩行者２)
}

Agt_Step{
        Dim Zentai As Integer
        Dim stepsu As Integer

        World.残留歩行者数 = _CountAgent(World.街.歩行者)
        World.残留歩行者数２ = _CountAgent(World.街.歩行者２)

        zentai = World.残留歩行者数 + World.残留歩行者数２
        stepsu = _getcountstep()
//      _DebugStr(zentai)

        if zentai == 0 then
                _ExitSimulationMsg ( "ステップ数=" & stepsu )
        end if
}
```

A.1.2　エージェントプログラム

・道順を知っているエージェントのルール

```
Agt_Init{
        Dim k As Integer
        Dim dx as double
        Dim dy as double

        Dim m as double
        Dim minL as double

        my.知った = false
        my.目的地 = false
        my.きたことある = false
        My.速さ = Rnd()*1.0 + 1.0

/*************************************************************/
/***避難開始時にどこに避難するか(目標避難場所)を決定する***/
/*************************************************************/
        目的地決定()
}

Agt_Step{

        Dim 周りのエージェント As Integer
        Dim 左上 As Integer
        Dim 左 As Integer
        Dim 左下 As Integer
        Dim hantei As Integer

        Dim obj As Object
```

```
        If Abs(CInt(My.X) - CInt(My.目的地 X)) * Abs(CInt(My.Y) - CInt(My.
目的地 Y)) < 0.5 Then
                My.目的地 X = world.街.出口(my.目標避難場所).X
                My.目的地 Y = world.街.出口(my.目標避難場所).Y
        End If
//      World.二次元空間(My.目的地 X, My.目的地 Y).色 = 1

// 音が聞こえる場合
        if (my.x > 39) and (my.x <= 70) then
                my.知った = true
                周りのエージェント = _ViewCountAgent (My.X,
My.Y,1,World.街.障害物)
                If 周りのエージェント > 1 Then
                '混んでいるとき
        //      My.色 = COLOR_RED
                        _MoveToSpace (My.X, My.Y, my.速さ)
                        目的地決定()
                        My.目的地 X = world.街.出口(my.目標避難場所).X
                        My.目的地 Y = world.街.出口(my.目標避難場所).Y
                        My.DIRECTION = 目的地の方向()
                        _forward(My.速さ)

                        左上 = _ViewCountAgent (My.X, My.Y -
1,0,World.街.障害物)
                        左 = _ViewCountAgent (My.X, My.Y ,0,World.
街.障害物)
                        左下 = _ViewCountAgent (My.X, My.Y +
1,0,World.街.障害物)

        //              if ((左上+左+左下) == 3) and (my.y <= 38) and
(my.y >= 20) then
```

```
//                          My.DIRECTION = 180
//                          _forward(My.速さ)
//                  end if

                    if (左上+左+左下)==3 then
                          My.DIRECTION = 0
                          _forward(My.速さ)
                    end if

                    if 左 == 1 and (左上 ==0 or 左下 == 0) then
                          My.DIRECTION = 90
                          _forward(My.速さ)
                    end if

              Else
              '混んでいないとき
//            My.色 = COLOR_BL
                    My.DIRECTION = 目的地の方向()
                    _forward(My.速さ)
              End If
//音が聞こえない場合

//上下方向////////////////////////////////////////////////////////////////////////////////////
        elseif ((my.y >= 9) and (my.y <= 33)) or ((my.y >= 38) and (my.y <=
62)) or ((my.y >= 67) and (my.y <= 90)) then
              if my.目的地 == true then
                    _forward(My.速さ)
              else
//                  My.色 = COLOR_BLUE
                    my.direction = 180 * Int(rnd()*2)          //上か下
か
                    _forward(My.速さ)
```

```
                    my.目的地 = true
                end if
//左方向////////////////////////////////////////////////////////////////////////////////
        elseif (((my.y >= 4) and (my.y <= 9)) or ((my.y >= 33) and (my.y <=
38)) or ((my.y >= 62) and (my.y <= 67)) or ((my.y >= 90) and (my.y <= 94)))
and (my.x >= 88)then
                my.direction =  90                          //左へ進
む
                my.目的地 = true
        //      My.色 = COLOR_yellow
                _forward(My.速さ)
//上下左右方向////////////////////////////////////////////////////////////////////////////////
        elseif (((my.y >= 33) and (my.y <= 38)) or ((my.y >= 62) and (my.y <=
67))) and ((my.x >= 71) and (my.x <= 76))then
                if (my.目的地 == true) and ((my.direction == 0) or
(my.direction == 180)) then        //上下からきた場合
                    hantei = Int(rnd()*2)
                    if my.きたことある == true then              //1
回来たことある場合
                        my.direction =  90
                        _forward(My.速さ)

                    elseif hantei == 1 then          //初めてきた場合
                        my.direction =  90 + 180 * Int(rnd()*2)
//右か左か
        //              My.色 = COLOR_BLACK
                        _forward(My.速さ)
                    else
                        _forward(My.速さ)
                    end if
                end if
                //左からきた場合////////////////////////////////////////////////////////////
```

```
                if (my.目的地 == true) and (my.direction == 90) then
                        hantei = Int(rnd()*2)
                        if my.きたことある == true then                 //1 回
来たことある場合
                                _forward(My.速さ)
                        elseif hantei == 1 then         //初めてきた場合
        //                      My.色 = COLOR_BLUE
                                my.direction = 180 * Int(rnd()*2)
//上か下か
                                my.きたことある = true
                                _forward(My.速さ)
                        else
                                _forward(My.速さ)
                                my.きたことある = true
                        end if
                end if
                //進行方向が右の場合//////////////////////////////////////////////////////////////
                if (my.目的地 == true) and (my.direction == 270) then
                        _forward(My.速さ)
                end if
                //初期値の場合//////////////////////////////////////////////////////////////
                if my.目的地 == false then
                        my.direction = 360 * Int(rnd()*4)/4                 //
上下左右
                        my.目的地 = true
                //      My.色 = COLOR_green
                        _forward(My.速さ)
                end if
//その他の方向//////////////////////////////////////////////////////////////
        else
                my.きたことある = true
                if (my.目的地 == false) or (my.direction == 0) or
```

```
(my.direction == 180) then
                        my.direction =   90 + 180 * Int(rnd()*2)          //
右か左か
                //      My.色 = COLOR_BLACK
                        my.目的地 = true
                        _forward(My.速さ)
                else
                        _forward(My.速さ)
                end if
        end if

        If ((my.X <= 39) and (my.Y >= 9)) or ((my.X <= 39) and (my.Y <= 60))
Then
                _KillAgent (My)
        End If
}


/*************************************************************/
/***避難開始時にどこに避難するか(目標避難場所)を決定する***/
/*************************************************************/
Function 目的地決定() As Double
{
        Dim k As Integer
        Dim dx as double
        Dim dy as double

        Dim m as double
        Dim minL as double

//すべての避難場所について距離を計算する
        for k = 0 to 51
```

```
                dx = world.街.出口(k).X - my.X
                dy = world.街.出口(k).Y - my.Y
                my.避難場所までの距離(k) = SQR(dx^2 + dy^2)
                /*_DebugStr(my.避難場所までの距離(k))*/
        next k


//最も近い避難場所を目標避難場所とする
        minL = my.避難場所までの距離(0)
        /*_DebugStr(minL)*/
        for m = 0 to 51
                if minL >= my.避難場所までの距離(m) then
                        minL = my.避難場所までの距離(m)
                        my.目標避難場所 = m
                end if
        next m

        Return(my.目標避難場所)

}

/*
        自分の位置から目的地への方向を絶対角度で返す。
*/
Function 目的地の方向() As Double
{
        Dim rtn As Double

        If (My.X < My.目的地 X) AND (My.目的地 Y < My.Y) Then
                rtn = 270 + _RadToDegree(Atn((my.Y - my.目的地 Y)/(my.
目的地 X - my.X)))
        End If
```

```
        If (My.X == My.目的地 X) AND (My.目的地 Y < My.Y) Then
                rtn = 0
        End If
        If (My.目的地 X < My.X) AND (My.目的地 Y < My.Y) Then
                rtn = 90 - _RadToDegree(Atn((my.Y - my.目的地 Y)/(my.X -
my.目的地 X)))
        End If
        If (My.目的地 X < My.X) AND (My.目的地 Y == My.Y) Then
                rtn = 90
        End If
        If (My.目的地 X < My.X) AND (My.Y < My.目的地 Y) Then
                rtn = 90 + _RadToDegree(Atn((my.目的地 Y - my.Y)/(my.X -
my.目的地 X)))
        End If
        If (My.目的地 X == My.X) AND (My.Y < My.目的地 Y) Then
                rtn = 180
        End If
        If (My.X < My.目的地 X) AND (My.Y < My.目的地 Y) Then
                rtn = 270 - _RadToDegree(Atn((my.目的地 Y - my.Y)/(my.目
的地 X - my.X)))
        End If
        If (My.X < My.目的地 X) AND (My.Y == My.目的地 Y) Then
                rtn = 270
        End If

        Return(rtn )
}
```

・道順を知らないエージェントのルール

```
Agt_Init{
        Dim k As Integer
        Dim dx as double
        Dim dy as double

        Dim m as double
        Dim minL as double

        My.速さ = Rnd()*1.0 + 1.0
        my.知った = false
        my.きた数=0
        my.きた数２=0
}

Agt_Step{

        Dim 周りのエージェント As Integer
        Dim 周りの歩行者 As Integer
        Dim 左上 As Integer
        Dim 左 As Integer
        Dim 左下 As Integer

        Dim hantei As Integer

        If Abs(CInt(My.X) - CInt(My.目的地 X)) * Abs(CInt(My.Y) - CInt(My.
目的地 Y)) < 0.5 Then
                My.目的地 X = world.街.出口(my.目標避難場所).X
                My.目的地 Y = world.街.出口(my.目標避難場所).Y
        End If
//      World.二次元空間(My.目的地 X, My.目的地 Y).色 = 1
```

```
        周りの歩行者 = _ViewCountAgent (My.X, My.Y,1,World.街.歩行者)
//周りに道を知ってる人がいる場合
        If ((周りの歩行者 >= 1) and (my.x <= 70)) or (my.x <= 42) Then
                my.知った = true
        end if

        if my.知った == true then
                My.色 = COLOR_RED
                周りのエージェント = _ViewCountAgent (My.X,
My.Y,1,World.街.障害物)
        //      _DebugStr(_ViewCountAgent (My.X, My.Y,1,World.街.障害
物))
                If 周りのエージェント > 1 Then
                '混んでいるとき
                        _MoveToSpace (My.X, My.Y, my.速さ)
                        目的地決定()
                        My.目的地 X = world.街.出口(my.目標避難場所).X
                        My.目的地 Y = world.街.出口(my.目標避難場所).Y
                        My.DIRECTION = 目的地の方向()
                        _forward(My.速さ)

                        左上 = _ViewCountAgent (My.X, My.Y -
1,0,World.街.障害物)
                        左 = _ViewCountAgent (My.X, My.Y ,0,World.
街.障害物)
                        左下 = _ViewCountAgent (My.X, My.Y +
1,0,World.街.障害物)

                        if (左上+左+左下)==3 then
                                My.DIRECTION = 0
                                _forward(My.速さ)
```

```
                    end if

                    if 左 == 1 and (左上 ==0 or 左下 == 0) then
                        My.DIRECTION = 90
                        _forward(My.速さ)
                    end if
                Else
                '混んでいないとき
        //      My.色 = COLOR_BL
                    My.DIRECTION = 目的地の方向()
                    _forward(My.速さ)
                End If
//周りに道を知ってる人がいない場合
//上下方向////////////////////////////////////////////////////////////////////////////////
        elseif ((my.y >= 9) and (my.y <= 33)) or ((my.y >= 38) and (my.y <=
62)) or ((my.y >= 67) and (my.y <= 90)) then
                if my.目的地 == true then
                    _forward(My.速さ)
                else
                //      My.色 = COLOR_BLUE
                    my.direction = 180 * Int(rnd()*2)          //上か下
か
                    _forward(My.速さ)
                    my.目的地 = true
                end if
//左方向////////////////////////////////////////////////////////////////////////////////
        elseif (((my.y >= 4) and (my.y <= 9)) or ((my.y >= 33) and (my.y <=
38)) or ((my.y >= 62) and (my.y <= 67)) or ((my.y >= 90) and (my.y <= 94)))
and (my.x >= 88)then
                my.direction =  90                                  //左へ進
む
                my.目的地 = true
```

```
        //          My.色 = COLOR_yellow
                    _forward(My.速さ)
//左右方向////////////////////////////////////////////////////////////////////////////////
        elseif ((my.x >= 54) and (my.x <= 59)) or ((my.x >= 71) and (my.x <=
76)) then
                if ((my.y >= 4) and (my.y <= 9)) or ((my.y >= 90) and (my.y
<= 94)) then
                        hantei = Int(rnd()*2)
                        my.きた数 = my.きた数 + 1
//1 回来たことある場合
                        if my.きた数 >= 6 then
                                my.direction =  90
                                _forward(My.速さ)
//初めてきた場合
                        elseif my.目的地 == false then
                      //初めからいた場合
                                        my.direction =  180 + 90 *
Int(rnd()*2)        //下か右か
                                        _forward(My.速さ)
                                        my.目的地 = true
                        elseif (hantei == 1) and (my.direction == 270)
then            //左から来た
                                if (my.y >= 4) and (my.y <= 9) then
//上にいるとき
                                        my.direction =  180         //
下へ
                                        _forward(My.速さ)
                                end if
                                if (my.y >= 90) and (my.y <= 94) then
//下にいるとき
                                        my.direction =  0           //上
へ
```

```
                                        _forward(My.速さ)
                                end if
                        elseif (hantei == 1) and (my.direction == 0) then
//上からきた
                                my.direction =  90 + 180 * Int(rnd()*2)
//右か左か
                                _forward(My.速さ)
                        elseif (hantei == 1) and (my.direction == 180)
then            //下からきた
                                my.direction =  90 + 180 * Int(rnd()*2)
//右か左か
                                _forward(My.速さ)
                        elseif (hantei == 1) and (my.direction == 90) then
//右から来た
                                if (my.y >= 4) and (my.y <= 9) then
//上にいるとき
                                        my.direction =  180          //
下へ
                                        _forward(My.速さ)
                                end if
                                if (my.y >= 90) and (my.y <= 94) then
//下にいるとき
                                        my.direction =  0          //上
へ
                                        _forward(My.速さ)
                                end if

                        else
                                _forward(My.速さ)
                        end if
//上下左右方向////////////////////////////////////////////////////////////////////////////////
                else
```

```
//上下からきた場合
                    if (my.目的地 == true) and ((my.direction == 0) or
(my.direction == 180)) then
                        hantei = Int(rnd()*2)
                        my.きた数２ = my.きた数２ + 1
                        if my.きた数２ >= 6 then          //1
回来たことある場合
                            my.direction =  90
                            _forward(My.速さ)

                        elseif hantei == 1 then          //初めてき
た場合
                            my.direction =  90 + 180 *
Int(rnd()*2)          //右か左か
                    //          My.色 = COLOR_BLACK
                            _forward(My.速さ)
                        else
                            _forward(My.速さ)
                        end if
//右からきた場合
                    elseif (my.目的地 == true) and (my.direction ==
90) then
                        hantei = Int(rnd()*2)
                        if my.きた数２ >= 6 then          //1
回来たことある場合
                            _forward(My.速さ)
                        elseif hantei == 1 then          //初めてき
た場合
                    //          My.色 = COLOR_BLUE
                            my.direction = 180 *
Int(rnd()*2)          //上か下か
                            _forward(My.速さ)
```

```
                                    else
                                        _forward(My.速さ)
                                    end if
//左から来た場合
                        elseif (my.目的地 == true) and (my.direction ==
270) then
                                    hantei = Int(rnd()*2)
                                    if my.きた数２ >= 6 then          //1
回来たことある場合
                                        my.direction = 90
                                        _forward(My.速さ)
                                    elseif hantei == 1 then       //初めてき
た場合
                        //              My.色 = COLOR_BLUE
                                        my.direction    =    180    *
Int(rnd()*2)        //上か下か
                                        _forward(My.速さ)
                                    else
                                        _forward(My.速さ)
                                    end if
//初期値の場合
                        else
                                    my.direction  =  360  *  Int(rnd()*4)/4
//上下左右
                                    my.目的地 = true
                        //          My.色 = COLOR_green
                                    _forward(My.速さ)
                        end if
                end if
//その他の左右方向////////////////////////////////////////////////////////////////////////////////
        else
                if my.目的地 == false then
```

```
                    my.direction =   90 + 180 * Int(rnd()*2)          //
右か左か
                    my.目的地 = true
                //          My.色 = COLOR_green
                    _forward(My.速さ)
                else
                    _forward(My.速さ)
                end if
        end if
//避難完了////////////////////////////////////////////////////////////////////////////////////////
        If (my.X <= 39) and (my.Y <= 60) Then   //and (my.Y >= 9)
            _KillAgent (My)
        End If
}


/*************************************************************/
/***避難開始時にどこに避難するか(目標避難場所)を決定する***/
/*************************************************************/
Function 目的地決定() As Double
{
        Dim k As Integer
        Dim dx as double
        Dim dy as double

        Dim m as double
        Dim minL as double

//すべての避難場所について距離を計算する
        for k = 0 to 51
            dx = world.街.出口(k).X - my.X
            dy = world.街.出口(k).Y - my.Y
```

```
                my.避難場所までの距離(k) = SQR(dx^2 + dy^2)
                /*_DebugStr(my.避難場所までの距離(k))*/
        next k


//最も近い避難場所を目標避難場所とする
        minL = my.避難場所までの距離(0)
        /*_DebugStr(minL)*/
        for m = 0 to 51
                if minL >= my.避難場所までの距離(m) then
                        minL = my.避難場所までの距離(m)
                        my.目標避難場所 = m
                end if
        next m

        Return(my.目標避難場所)

}

/*
        自分の位置から目的地への方向を絶対角度で返す。
*/
Function 目的地の方向() As Double
{
        Dim rtn As Double

        If (My.X < My.目的地 X) AND (My.目的地 Y < My.Y) Then
                rtn = 270 + _RadToDegree(Atn((my.Y - my.目的地 Y)/(my.
目的地 X - my.X)))
        End If
        If (My.X == My.目的地 X) AND (My.目的地 Y < My.Y) Then
                rtn = 0
```

```
        End If
        If (My.目的地 X < My.X) AND (My.目的地 Y < My.Y) Then
                rtn = 90 - _RadToDegree(Atn((my.Y - my.目的地 Y)/(my.X -
my.目的地 X)))
        End If
        If (My.目的地 X < My.X) AND (My.目的地 Y == My.Y) Then
                rtn = 90
        End If
        If (My.目的地 X < My.X) AND (My.Y < My.目的地 Y) Then
                rtn = 90 + _RadToDegree(Atn((my.目的地 Y - my.Y)/(my.X -
my.目的地 X)))
        End If
        If (My.目的地 X == My.X) AND (My.Y < My.目的地 Y) Then
                rtn = 180
        End If
        If (My.X < My.目的地 X) AND (My.Y < My.目的地 Y) Then
                rtn = 270 - _RadToDegree(Atn((my.目的地 Y - my.Y)/(my.目
的地 X - my.X)))
        End If
        If (My.X < My.目的地 X) AND (My.Y == My.目的地 Y) Then
                rtn = 270
        End If

        Return(rtn )
}
```

A.2 パラメトリックスピーカを使用した場合の避難誘導

A.2.1 ワールドルール

```
Agt_Init{
        Dim i as integer
        For i = 0 to WORLD.人の数 - 1
        _CreateAgent(WORLD.街.歩行者)
        Next i
        _RandomPutAgent(WORLD.街.歩行者, false)

        World.残留歩行者数 = _CountAgent(World.街.歩行者)
}

Agt_Step{
        Dim stepsu as integer
        World.残留歩行者数 = _CountAgent(World.街.歩行者)
        stepsu = _getcountstep()
//      _DebugStr(zentai)

        if World.残留歩行者数 == 0 then
                _ExitSimulationMsg ( "ステップ数=" & stepsu )
        end if

}
```

A.2.2 エージェントルール

```
Agt_Init{
        Dim k As Integer
        Dim dx as double
        Dim dy as double

        Dim m as double
        Dim minL as double

        My.速さ = Rnd()*1.0 + 1.0
}

Agt_Step{

        Dim 周りのエージェント As Integer
        Dim 左上 As Integer
        Dim 左 As Integer
        Dim 左下 As Integer
        Dim obj As Object

        if (my.y >= 4) and (my.y <= 9) then
                if (my.x >= 37) and (my.x <= 42) then
                        目的地決定()
                        My.目的地 X = world.街.出口(my.目標避難場所).X
                        My.目的地 Y = world.街.出口(my.目標避難場所).Y
                        My.DIRECTION = 目的地の方向()
                        ぶつかり阻止()
                else
                        my.direction = 90
                        ぶつかり阻止()
                end if
```

```
        end if

        if (my.y > 9) and (my.y <= 33) then
                if (my.x >= 37) and (my.x <= 42) then
                        目的地決定()
                        My.目的地 X = world.街.出口(my.目標避難場所).X
                        My.目的地 Y = world.街.出口(my.目標避難場所).Y
                        My.DIRECTION = 目的地の方向()
                        ぶつかり阻止()
                else
                        my.direction = 0
                        ぶつかり阻止()
                end if
        end if
/*
        if (my.y > 21) and (my.y <= 33) then
                if (my.x >= 37) and (my.x <= 42) then
                        目的地決定()
                        My.目的地 X = world.街.出口(my.目標避難場所).X
                        My.目的地 Y = world.街.出口(my.目標避難場所).Y
                        My.DIRECTION = 目的地の方向()
                        ぶつかり阻止()
                else
                        my.direction = 180
                        ぶつかり阻止()
                end if
        end if
*/
        if (my.y > 33) and (my.y <= 38) then
                if (my.x >= 37) and (my.x <= 42) then
                        目的地決定()
                        My.目的地 X = world.街.出口(my.目標避難場所).X
```

```
                My.目的地 Y = world.街.出口(my.目標避難場所).Y
                My.DIRECTION = 目的地の方向()
                ぶつかり阻止()
        else
                my.direction = 90
                ぶつかり阻止()
        end if
end if

if (my.y > 38) and (my.y <= 62) then
        if (my.x >= 37) and (my.x <= 42) then
                目的地決定()
                My.目的地 X = world.街.出口(my.目標避難場所).X
                My.目的地 Y = world.街.出口(my.目標避難場所).Y
                My.DIRECTION = 目的地の方向()
                ぶつかり阻止()
        else
                my.direction = 0
                ぶつかり阻止()
        end if
end if

if (my.y > 62) and (my.y <= 67) then
        if (my.x >= 37) and (my.x <= 42) then
                目的地決定()
                My.目的地 X = world.街.出口(my.目標避難場所).X
                My.目的地 Y = world.街.出口(my.目標避難場所).Y
                My.DIRECTION = 目的地の方向()
                ぶつかり阻止()
        else
                my.direction = 90
                ぶつかり阻止()
```

```
        end if
end if

if (my.y > 67) and (my.y <= 90) then
        if (my.x >= 37) and (my.x <= 42) then
                目的地決定()
                My.目的地 X = world.街.出口(my.目標避難場所).X
                My.目的地 Y = world.街.出口(my.目標避難場所).Y
                My.DIRECTION = 目的地の方向()
                ぶつかり阻止()
        else
                my.direction = 0
                ぶつかり阻止()
        end if
end if

if (my.y > 90) and (my.y <= 94) then
        if (my.x >= 37) and (my.x <= 42) then
                目的地決定()
                My.目的地 X = world.街.出口(my.目標避難場所).X
                My.目的地 Y = world.街.出口(my.目標避難場所).Y
                My.DIRECTION = 目的地の方向()
                ぶつかり阻止()
        else
                my.direction = 90
                ぶつかり阻止()
        end if
end if

If ((my.X <= 39) and (my.Y <= 60)) Then
        _KillAgent (My)
End If
```

```
}

/*************************************************************/
/***避難開始時にどこに避難するか(目標避難場所)を決定する***/
/*************************************************************/
Function 目的地決定() As Double
{
        Dim k As Integer
        Dim dx as double
        Dim dy as double

        Dim m as double
        Dim minL as double

//すべての避難場所について距離を計算する
        for k = 0 to 17
                dx = world.街.出口(k).X - my.X
                dy = world.街.出口(k).Y - my.Y
                my.避難場所までの距離(k) = SQR(dx^2 + dy^2)
                /*_DebugStr(my.避難場所までの距離(k))*/
        next k


//最も近い避難場所を目標避難場所とする
        minL = my.避難場所までの距離(0)
        /*_DebugStr(minL)*/
        for m = 0 to 17
                if minL >= my.避難場所までの距離(m) then
                        minL = my.避難場所までの距離(m)
                        my.目標避難場所 = m
                end if
        next m
```

```
	Return(my.目標避難場所)

}

/*
	自分の位置から目的地への方向を絶対角度で返す。
*/
Function 目的地の方向() As Double
{
	Dim rtn As Double

	If (My.X < My.目的地 X) AND (My.目的地 Y < My.Y) Then
		rtn = 270 + _RadToDegree(Atn((my.Y - my.目的地 Y)/(my.
目的地 X - my.X)))
	End If
	If (My.X == My.目的地 X) AND (My.目的地 Y < My.Y) Then
		rtn = 0
	End If
	If (My.目的地 X < My.X) AND (My.目的地 Y < My.Y) Then
		rtn = 90 - _RadToDegree(Atn((my.Y - my.目的地 Y)/(my.X -
my.目的地 X)))
	End If
	If (My.目的地 X < My.X) AND (My.目的地 Y == My.Y) Then
		rtn = 90
	End If
	If (My.目的地 X < My.X) AND (My.Y < My.目的地 Y) Then
		rtn = 90 + _RadToDegree(Atn((my.目的地 Y - my.Y)/(my.X -
my.目的地 X)))
	End If
	If (My.目的地 X == My.X) AND (My.Y < My.目的地 Y) Then
		rtn = 180
```

```
        End If
        If (My.X < My.目的地 X) AND (My.Y < My.目的地 Y) Then
                rtn = 270 - _RadToDegree(Atn((my.目的地 Y - my.Y)/(my.目
的地 X - my.X)))
        End If
        If (My.X < My.目的地 X) AND (My.Y == My.目的地 Y) Then
                rtn = 270
        End If

        Return(rtn )
}

sub ぶつかり阻止()
{
        Dim 周りのエージェント As Integer
        周りのエージェント = _ViewCountAgent (My.X, My.Y,1,World.街.障
害物)

        If 周りのエージェント > 1 Then
                '混んでいるとき
                My.色 = COLOR_green
                _MoveToSpace (My.X, My.Y, my.速さ)
        else
                My.色 = COLOR_red
                _forward(My.速さ)
        end if
}
```