ВОССТАНОВЛЕНИЕ АФФИННОЙ СТРУКТУРЫ СЦЕНЫ ПО ДВИЖЕНИЮ

Афиногенов Е.И.
Научный руководитель: к.т.н, доцент Волосатова Т.М.
Кафедра «САПР» (РК6) МГТУ им. Н.Э.Баумана, Москва, Россия

RECONSTRUCTION OF AFFINE STRUCTURE ON MOTION

Afinogenov E.I.
Scientific chief: PhD, associate professor Volosatova T.M.
CAD/CAE Department (RK6) of BMSTU, Moscow, Russia

Аннотация
Рассматривается задача восстановления аффинной структуры сцены по ряду изображений. При этом делается допущение о наличии только согласованных проекций точек сцены. Для решения задачи предлагается использовать устойчивую схему факторизации на основе разложения по сингулярным значениям.

Abstract
This article deals with reconstruction of affine structure of scene from set of pictures. It is assumed that projects of points are correspondent. Singular Value Decomposition method is proposed for solving such problem with least squares scheme.

В системах 3D компьютерного зрения часто для получения трехмерного представления сцены используется механизм стереореконструкции по двум изображениям. При этом камеры, используемые для получения входных изображений, обычно калибруются так, что их внутренние параметры известны, а внешние определены относительно некоторой глобальной системы отсчета. В данной работе рассматривается более сложный случай, когда положения камер и, возможно, их внутренние параметры не известны априори и могут меняться со временем. Такая ситуация встречается в приложениях *визуализации на основе анализа изображений*, а также в *системах активного зрения*, калибровочные параметры которых меняются в процессе работы.

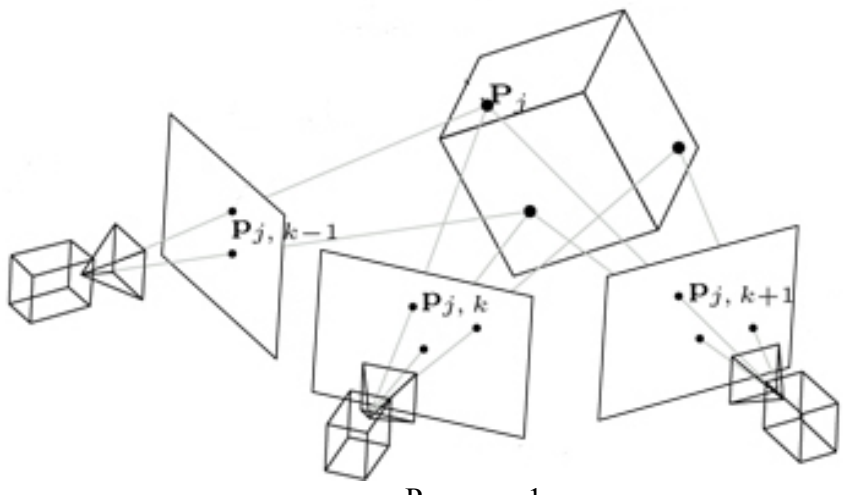


Рисунок 1

В данной работе не рассматривается задача определения соответствий. Будем считать, что имеется ***m*** согласованных проекций ***n*** точек. Для удобства все вычисления будем проводить в однородных координатах.

При этом будем рассматривать сцены, рельеф которых меняется незначительно по сравнению с глубиной по отношению к камерам. Тогда перспективу можно аппроксимировать простыми аффинными моделями [1].

Аффинные проекции p_{ij} точек P_j можно записать следующим образом:

$$p_{ij} = M_i \begin{pmatrix} P_j \\ 1 \end{pmatrix} = A_i P_j + b_i \qquad i = 1,...,m \qquad j = 1,...,n \qquad (1)$$

Таким образом, *определение аффинной структуры по движению* – это задача оценки $\boldsymbol{m}$ матриц $M_i = \begin{pmatrix} A_i & b_i \end{pmatrix}$ размера 2x4 и $\boldsymbol{n}$ положений точек P_j в некоторой глобальной системе координат по $\boldsymbol{mn}$ соответствиям элементов изображения p_{ij}.

Важно понимать, что если M_i и P_j – это решения уравнения (1), то решениями также являются M'_i и P'_j такие, что

$$M'_i = M_i Q \qquad (2)$$

$$\begin{pmatrix} P'_j \\ 1 \end{pmatrix} = Q^{-1} \begin{pmatrix} P_j \\ 1 \end{pmatrix} \qquad (3)$$

То есть любое решение задачи нахождения аффинной структуры по движению можно определить только с точностью до аффинного преобразования.

Пусть статичная сцена наблюдается с помощью фиксированного набора $\boldsymbol{m}$ аффинных камер. Через $\boldsymbol{p_1,\ldots,p_m}$ обозначим $\boldsymbol{m}$ проекций точки сцены P.

$$q = r + AP \qquad (4)$$

где согласно (1):

$$q = \begin{pmatrix} p_1 \\ ... \\ p_m \end{pmatrix} \qquad r = \begin{pmatrix} b_1 \\ ... \\ b_m \end{pmatrix} \qquad A = \begin{pmatrix} A_1 \\ ... \\ A_m \end{pmatrix} \qquad (5)$$

При рассмотрении $\boldsymbol{n}$ точек $P_1,...,P_n$ можно определить матрицу данных D:

$$D = \begin{pmatrix} q_1 & ... & q_n \\ 1 & ... & 1 \end{pmatrix} \qquad (6)$$

В работе [3] для решения задачи предлагается использовать устойчивую схему факторизации на основе разложения по сингулярным значениям (SVD – Singular Value Decomposition). Кратко алгоритм определения аффинной формы выглядит следующим образом:

1. Найти SVD-представление матрицы $D = UWV^T$
2. Построить матрицы U_3, V_3 и W_3 из трех левых крайних столбцов соответствующих матриц U, V и W.
3. Определить

$A_0 = U_3$ – оценка движения камеры.

$P_0 = W_3 V^T{}_3$ – оценка структуры сцены.

Предположим, что жесткий объект наблюдается откалиброванными ортографическими камерами, а точки изображений представляются их нормированными координатными векторами. В этом случае преобразование между координатными системами, связанными с камерами, из аффинного становится евклидовым, то есть можно записать как суперпозицию вращения и трансляции.

Известно, что все решения задачи определения структуры по движению равны с точностью до аффинной неопределенности. Если положение точки объекта в евклидовой системе координат равно $\hat{P}$, а соответствующая проекционная матрица имеет вид $\hat{M} = (\hat{A} \quad \hat{b})$, то должно существовать некоторое аффинное преобразование Q:

$$Q = \begin{pmatrix} C & d \\ 0^T & 1 \end{pmatrix} \qquad (7)$$

такое, что $\hat{M} = MQ$ и $\hat{P} = C^{-1}(P - d)$.

Подобное преобразование называется *евклидовым уточнением*, поскольку оно отображает аффинную форму в евклидову.

При наличии $m > 2$ изображений данную переопределенную систему уравнений можно решить с помощью метода наименьших квадратов [2].

Литература

1. Форсайт Д., Понс Ж. Компьютерное зрение. Современный подход. : Пер. с англ. – М.: Издательский дом «Вильямс», 2004. – 928 с.
2. Шор. Я. Б. Статистические методы анализа и контроля качества и надежности. М.:Госэнергоиздат, 1962, с. 552, С. 92-98
3. Tomasi C., Kanade T. Shape and Motion from Image Streams: a Factorization Method Full Report on the Orthographic Case, 1992
4. Golub G.H., Reinsch C. Singular value decomposition and least squares solutions, In Handbook for Automatic Computation, volume 2, pages 134-151. Springer Verlag, 1971.
5. Lucas B.D., Kanade T. An iterative image registration technique with an application to stereo vision. In Proceedings of the 7th International Joint Conference on Artificial Intelligence, 1981.

ОЦЕНКА ПОЖАРНОЙ ОБСТАНОВКИ Г. МОСКВЫ

Наумова А.А., Павлов Ю.Н.
МГТУ им. Н.Э.Баумана, Москва, Россия

RECONSTRUCTION OF AFFINE STRUCTURE ON MOTION

Naumova A.A., Pavlov J.N.
BMSTU, Moscow, Russia

Аннотация
Рассматривается задача оценки пожарной безопасности г.Москвы.

Введение

Московская противопожарная служба резко отличается от других управлений ГПС, поскольку Москва является мегаполисом: здесь территория, и плотность населения, и количество высотных зданий гораздо больше, чем в любом другом городе России. К тому же в столице сконцентрированы различные объекты федерального значения. Поэтому задачи, которые приходится решать московским пожарным, достаточно серьезны.

Целями данной работы являлось изучение пожарной обстановки в г.Москве, анализ расположения пожарный частей, сопоставление расположения пожарных частей и количества возгораний в отдельных районах, оценка рациональности расположения пожарных частей и составление рекомендаций по улучшению противопожарной обстановки.

Структура пожарной службы г. Москвы

Структура пожарной службы г.Москвы аналогична структурам других управлений. В состав противопожарной службы входят основные и вспомогательные службы. К основным службам относятся следующие: служба по пожаротушению, центр управления силами и средствами, отдел Государственного пожарного надзора, объектовые пожарные части (например, при Большом театре, Третьяковской галерее, Мосэнерго, на крупных предприятиях), отделы ресурсного обеспечения, финансовый и технический. К вспомогательным службам относятся пожарно-технический центр (ПТЦ) и учебный центр, где производится подготовка и переподготовка личного состава.

Кроме этих отделений имеются окружные управления. Их расположение зависит от принадлежности к административным округам. В Москве 10 округов разной величины, соответственно имеется 10 окружных управлений. В этих управлениях тоже есть финансовые, технические отделы, отделения Госпожнадзора.

Завершающей ступенью в структуре УГПС являются пожарные оперативные части, которые непосредственно заняты выездами на пожары. По разным причинам одни части ликвидируются, другие создаются. Это связано с передислокацией и старением зданий и всей базы частей. Главная наша задача - избежать гибели людей на пожарах, и поэтому пожарные должны прибыть вовремя. Среднее время прибытия пожарных на место - 6,5 минут. В часы пик это время сильно увеличивается. Поэтому месторасположение каждой перемещаемой или новой пожарной части щепетильно продумывается.

Ситуация с пожарами в целом нормальная. Если мы будем сравнивать с 1990-ми годами, то число пожаров сократилось примерно на 10 тыс. В 1994-1995 гг. происходило в среднем 25 тыс. пожаров в год, сейчас их количество сократилось. Пожары в Подмосковье на количестве пожаров в самой Москве не отразились. Конечно, в засушливое лето довольно часто происходит возгорание мусора, но пожарами это крайне не желательно назвать. К тому же сейчас благодаря городским властям Москва стала чище. Раньше возгораний мусора было больше: горели свалки. Карта плотности возгораний (рис.1).

В данной работе были использованы материалы Управления по обеспечению мероприятий гражданской защиты города Москвы. Была создана карта плотности возгораний, основанная на статистических данных о количествах пожаров и возгораний в разных районах и округах Москвы.

Можно выделить области с резко повышенной плотностью:

1. Районы: Тверской, Мещанский.
2. Районы между ул. Б. Ордынка и Якиманка.
3. ВАО, между Щелковским шоссе и Измайловским бульваром. Некоторые области с коэффициенты среднего значения с повышенной плотностью:
4. Районы вдоль Бутырской ул. и Дмитровского шоссе.
5. Районы Ховрино, Головинский, Зап. Дегунино.
6. Районы между м. Академическая и м. Новые Черемушки
7. Районы между м. Рязанский пр-т и пл. Перово;
8. Районы между м. Домодедовская и м. Красногвардейская (интерполяция с шагом 2000 м.).

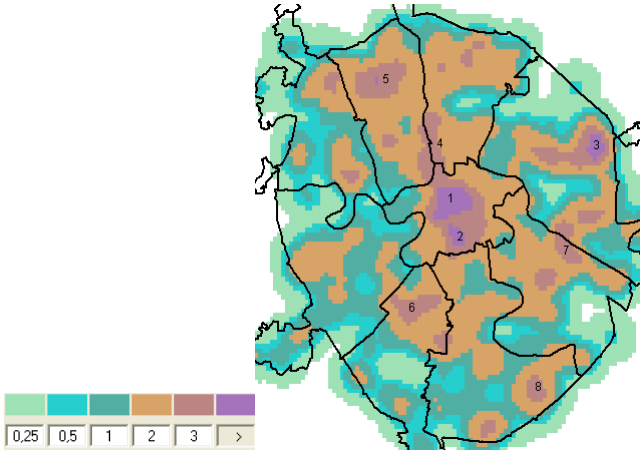


Рисунок 1 - Карта плотности пожаров за 2007-2008г.г.

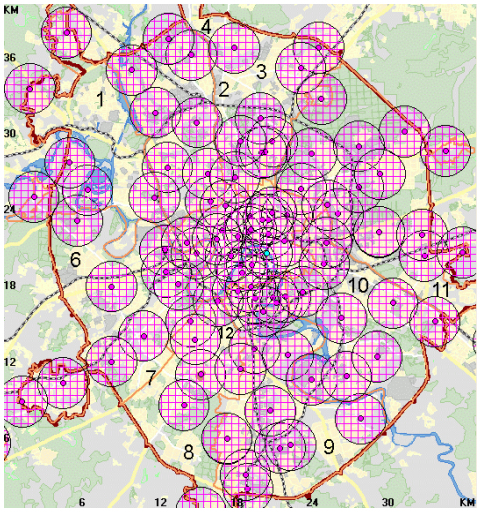


Рисунок 2 - Нормативные радиусы выезда подразделений пожарной охраны

Вероятнее всего, это связано с повышенной плотностью населения в этих областях. Большое количество пожаров в Центральном округе связано, вероятно, прежде всего с тем, что большинство зданий здесь преимущественно старой постройки с деревянными перекрытиями, что резко повышает вероятность возгорания

Однако взаимная близость пожарной части и потенциального места пожара в городе не является гарантией того, что пожарные расчеты смогут быстро подъехать к очагу возгорания, так как пожарные машины едут по дорогам, а в крупном городе расстояние по прямой и по дорогам (маршрут) в большинстве случаев не совпадают.

Нормативный радиус выезда подразделений пожарной охраны составляет 3 км по п.6 прил.1 СНиП 2.07.01-89*, а в Москве – 2км или 1км при высоте здания более 100м (согласно п.10.4 МГСН 1.04-2005) или в условиях недостаточного технического оснащения пожарными автолестницами (автоподъемниками) высотой 50м и более, а также автонасосами высокого давления.

Так как стометровых зданий (свыше 30 этажей) в Москве не так много, поэтому примем нормативный радиус выезда подразделений пожарной охраны, равным 2 км.

В соответствии с вышеизложенным на карту Москвы были нанесены круги соответствующие нормативному радиусу (2 км). В итоге получилось, что этими кругами покрыта почти вся территория города.

С другой стороны во всех округах, кроме ЦАО, эти окружности мало пересекаются, т.е. в случае серьезного пожара, требующего многих пожарных расчетов, соседние пожарные части не смогут достаточно быстро подъехать т.к. находятся за пределами нормативного радиуса. В результате анализа были выявлены районы не входящие ни в одну нормативную зону обслуживания: Тушино, Отрадное, Медведково, Угличская ул., Рябиновская ул, Молодежная, Ленинский пр-т, Ясенево, Орехово, Рязанский пр-т, Косино, Черемушки.

Некоторые области являются областями с наибольшей частотой пожаров, и, причем эти области совпадают с областями, для которых длина маршрута от ближайшей пожарной части более чем в двое превышает нормативную.Такими областями являются: Южное Медведково и Отрадное, район Аэропорт (Балтийская ул.), частично Западное и Восточное Бирюлево, Можайский район, районы Текстильщики, Рязанский и частично Нижегородский (в районе Волжского бульвара и Газгольдерной ул.), Тушино, Рябиновская ул., между м. Измайловская и м. Щелковская, Черемушки, между м. Домодедовская и м. Красногвардейская

Кроме того есть некоторые области с плотной застройкой и длиной маршрута значительно превышающей нормативную: Покровское-Стрешнево (ст. м. Тушинская и Щукинская), Кунцево, Можайский, Очаково-Матвеевская (в районе Рябиновой ул.), Ясенево, Ярославский р-н, между парком Лосиный остров и Ярославской ж.д.

Если рассмотреть в совокупности карты плотности пожаров и карты с нормативными радиусами выезда подразделений пожарной охраны, то можно проследить насколько оптимальным является существующее расположение пожарных частей по отношению к местам реально возникающих пожаров.

Если считать, что наиболее эффективным расположением пожарных частей является их размещение в областях наибольшей частоты возникновения пожаров, то такому критерию удовлетворяет только центральный округ. В остальных районах тенденция обратная.

Так сложилось скорее всего потому, что пожарные части в центре располагаются исторически, со времен старой Москвы, в то время как в новых районах они не вписываются в застройку современных кварталов.

Оценка релевантности данных

Также одним из рассмотренных вопросов был вопрос о релевантности имеющихся данных, т.е. насколько имеющиеся данные (2006 - 2008г.г.) релевантные при анализе ситуации для последующих интервалов времени. Иными словами, будут ли области города с повышенной частотой пожаров оставаться таковыми и в последующие годы.

Для проверки этого положения был вычислены коэффициенты корреляции результатов интерполяции (двумерных поверхностей) для пар разных периодов времени и с разным лагом интерполяции.

Существует несколько различных коэффициентов корреляции, к каждому из которых относится сказанное выше. Наиболее широко известен коэффициент корреляции Пирсона, характеризующий степень линейной зависимости между переменными. Он определяется, как

$$r = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2} \sqrt{\sum_i (y_i - \bar{y})^2}}$$

Результаты этого эксперимента сведены в таблицу корреляций (Табл.1).

Таблица 1-Таблица корреляций

	Коэффициент корреляции		
Шаг (км)	2006-2007	2006-2008	2007-2008
1	0,39	0,41	0,41
2	0,49	0,65	0,63
3	0,71	0,75	0,72
4	0,77	0,80	0,72
5	0,81	0,83	0,79
6	0,83	0,84	0,81

Из таблицы следует, что уже на расстоянии 3 км наблюдается значительная корреляция, что позволяет с достаточной достоверностью использовать эти данные, по крайней мере, на ближайшие несколько лет.

Заключение

В работе были изучены и обработаны статистические данные о пожарной обстановке в г. Москве и материалы с данными о расположении пожарных частей. Была проведена оценка пожароопасности районов города с выявлением областей с наиболее большим количеством возгораний. Эти данные сопоставлялись с данными по областям с наименьшим количество пожарных частей или наиболее отдаленными от них. Был составлен ряд рекомендаций по улучшению противопожарной обстановки в городе. Были предложены варианты мест строительства необходимых новых пожарных частей. Также сделаны рекомендации по усилению готовности и бдительности пожарных частей, находящихся в непосредственной близости с наиболее пожароопасными и удаленными от пожарных частей районами.

По результатам анализа расположения пожарных частей в г. Москве сделать вывод об их удовлетворительном расположении на территории города. Однако существуют пожароопасные районы и районы с большой плотностью населения, в которых необходим постоянный и серьезный контроль над пожароопасной обстановкой.

Литература

1. Статистический данные Управления по обеспечению мероприятий гражданской защиты города Москвы (сайт http://www.mos.ru).
2. Палов Ю.Н. "Регрессионный анализ в прикладной задаче идентификации", Труды МГТУ 1992 год.
3. Якуш С. Е., Эсманский Р. К. Анализ пожарных рисков. Часть I: Подходы и методы. Проблемы анализа риска, т. 6, № 3, с. 8—27, 2009 год.

НЕЙРОСЕТЕВАЯ ИДЕНТИФИКАЦИЯ ДИНАМИЧЕСКОЙ СИСТЕМЫ ТИПА ГЕКСАПОД

Литун Т.О.
д.ф.-м.н., профессор Карпенко А.П.
МГТУ им. Н.Э. Баумана, г. Москва, Россия

IDENTIFICATION OF HEXAPOD TYPE DYNAMIC SYSTEM USING NEURAL NETWORKS

Litun T.O.
Professor A.P. Karpenko
Bauman Moscow State Technical University, Moscow, Russia

Аннотация
Работа посвящена нейросетевой идентификации динамической системы типа гексапод с помощью сетей NARX.
Abstract
The paper is dedicated to identification of hexapod type dynamic system using NARX neural networks.

Введение. Методы идентификации, как методы построения математических моделей реальных динамических систем, подверженных неконтролируемым случайным воздействиям (стохастических систем), сегодня являются важной составной частью процесса решения задач управления. Для нейросетевой идентификации динамических систем чаще всего используют *NARX* нейронные сети [1]. Работа выполнена с использованием программного пакета *MathWorks Matlab c Neural Network Toolbox*.

Для уменьшения временных затрат на обучение и повышения качества аппроксимации модели механизма типа гексапод, в работе используется ансамбль из шести нейронных сетей схожей топологии. Каждая из сетей получает на вход сигнал из шести управляющих воздействий. Выходом сети является значение единственной обобщенной координаты. Подобный подход позволяет использовать сети с различным числом нейронов и скрытых слоев для идентификационной модели.

Постановка задачи. При аппроксимации динамической системы нейронной сетью для обеспечения процесса ее обучения необходимо наличие предварительно аналитически или экспериментально определенного массива входных и соответствующих им выходных значений. В рамках рассматриваемой задачи таковыми являются значения внешних сил $F_1, F_2, F_3, F_4, F_5, F_6$, вызывающие двойные, противоположные по знаку ступенчатые изменения обобщенных координат $x_B, y_B, z_B, \varphi_1, \varphi_2, \varphi_3$ [2].

Ступенчатые изменения обобщенных координат задаются функцией

$$x(t) = \begin{cases} \dfrac{w}{1+e^{u(t-s)}} + q, t \le 7; \\ w + q, 7 < t \le 9; \\ \dfrac{w}{1+e^{-u(t-s)}} + q, t > 9. \end{cases} \quad (1)$$

Здесь w – амплитуда ступенчатого воздействия, u – крутизна подъема сигнала в точке $t = s$; q - начальное значение $x(0)$. При некоторых значениях свободных параметров w, u, s, q вид функции (1) иллюстрирует рисунок 1.

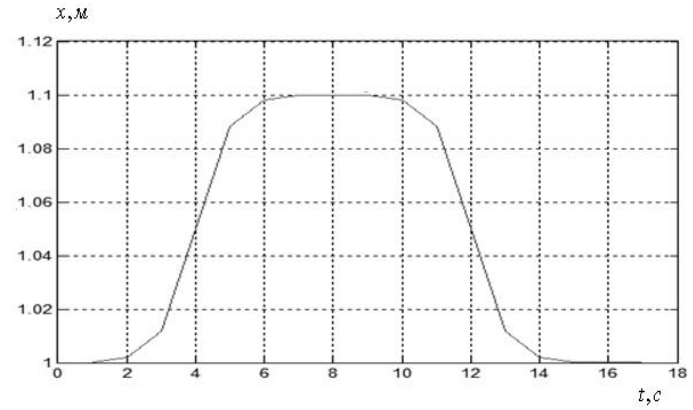


Рисунок 1 – К формированию обучающей выборки: вид функции (1)

В работе [2] показано, что динамика гексапода описывается системой обыкновенных дифференциальных уравнений (ОДУ). При формировании обучающей выборки в качестве функций $x_B(t)$, $y_B(t)$, $z_B(t)$, $\varphi_1(t)$, $\varphi_2(t)$, $\varphi_3(t)$ использовались функции вида (1). Для нахождения управляющих сил $F_1, F_2, F_3, F_4, F_5, F_6$, соответствующих указанным изменениям обобщенных координат, дважды дифференцируем функции $x_{B,i}(t)$, $y_{B,i}(t)$, $z_{B,i}(t)$, $\varphi_{1,i}(t)$, $\varphi_{2,i}(t)$, $\varphi_{3,i}(t)$, $i \in [1:10]$ по времени *t*. Полученное выражение имеет вид

$$a_{q,i} = \frac{-w_i u_i^2 e^{u_i(t-s_i)}(1-e^{u_i(t-s_i)})}{(1+e^{u_i(t-s_i)})^3} . \quad (2)$$

Подставив полученные выражения в систему ОДУ, описывающую динамику гексапода, для каждого момента времени *t* получим следующую систему линейных алгебраических уравнений (СЛАУ) относительно значений усилий $F_1, F_2, F_3, F_4, F_5, F_6$ в этот момент времени:

$$Ma_{x_B} = \sum_{i=1}^{6} F_i \cos\gamma_{i,1},\ M(a_{y_B} + g) = \sum_{i=1}^{6} F_i \cos\gamma_{i,2},\ Ma_{z_B} = \sum_{i=1}^{6} F_i \cos\gamma_{i,3}, \quad (3)$$

$$J_x a_{\varphi_1} = \sum_{i=1}^{6} F_i \sum_{j=1}^{3} \left(r_{i,j,1}(\varphi_1, \varphi_2, \varphi_3) \cos\gamma_{i,j} \right), \quad (4)$$

$$J_y a_{\varphi_2} = \sum_{i=1}^{6} F_i \sum_{j=1}^{3} \left(r_{i,j,2}(\varphi_1, \varphi_2, \varphi_3) \cos\gamma_{i,j} \right), \quad (5)$$

$$J_z a_{\varphi_3} = \sum_{i=1}^{6} F_i \sum_{j=1}^{3} \left(r_{i,j,3}(\varphi_1, \varphi_2, \varphi_3) \cos\gamma_{i,j} \right). \quad (6)$$

Совокупность обобщенных координат и соответствующих им силовых воздействий, составляет обучающую выборку, использованную при создании ансамбля нейронных сетей.

Существуют лишь общие рекомендации по выбору параметров нейронной сети. Применительно к рассматриваемой задаче моделирования требуется экспериментальное уточнение количества нейронов и скрытых слоев. Были обучены нейронные сети, параметры которых отражены в таблице 1.

Таблица 1 - Параметры используемых нейронных сетей

Нейронная сеть	Число скрытых слоев	Число нейронов в скрытом слое, N	Задержка по входу, *d*1	Задержка по выходу, *d*2
narx1	1	10	0	(1,1)
narx2	1	8	0	(1,1)
narx3	1	12	0	(1,1)
narx4	1	15	0	(1,1)

Эффективность нейросетевой аппроксимации модели гексапода иллюстрирует рисунок 2, на котором ε_Z, ε_{φ_2} - квадраты ошибок аппроксимации соответствующих обобщенных координат.

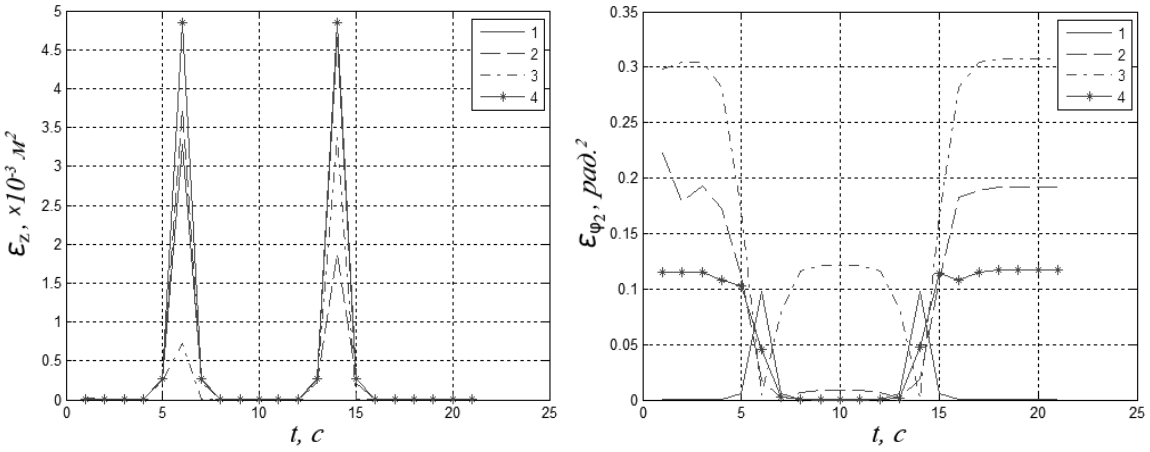


Рисунок 2 – Эффективность нейросетевой аппроксимации:
1 - *narx*1; 2 - *narx*2; 3 - *narx*3; 4 - *narx*4

В таблице 2 приведены результаты тестирования нейронных сетей. Здесь δ – усредненное значение квадрата ошибки. На входы сетей подавались сигналы, соответствующие управляющим силам, которые вызывают ступенчатые изменения обобщенных координат и возврат их в первоначальное состояние. Амплитуды «ступенек» находились в пределах 0,2 м для линейных и 1,05 рад для угловых обобщенных координат.

Таблица 2 показывает, что для обобщенных координат x_B, y_B, φ_1, φ_2, φ_3 меньшую ошибку аппроксимации обеспечивает сеть *narx*1, а для координаты z_B - сеть *narx*3.

Таблица 2 – Результаты тестирования нейронных сетей

название сети	Значения параметра δ					
	x_B, м^2	y_B, м^2	z_B, м^2	φ_1, м^2	φ_2, м^2	φ_3, м^2
narx1	0.0004	0.0007	0.0005	0.0017	0.0100	0.0067
narx2	0.0018	0.0009	0.0003	0.0030	0.0645	0.0071
narx3	0.0010	0.0008	0.0002	0.0046	0.1133	0.0069
narx4	0.0020	0.001	0.0005	0.0035	0.0414	0.0072

Для оценки качества обучения нейронных сетей, идентифицирующих динамическую систему типа гексапод, рассмотрены случаи, когда управляющие силовые воздействия вызывают изменения от одной до шести обобщенных координат. Некоторые результаты этого исследования приведены на рисунках 3, 4, где ψ – доля тестовых сигналов, отвечающих заданной точности аппроксимации (ошибка на фронте сигнала менее 30%), ζ – средняя квадратичная ошибка аппроксимации.

На рисунке 5 представлены примеры, иллюстрирующие точность нейросетевой идентификации рассматриваемого манипулятора параллельной кинематики.

Результаты исследования позволяют сделать вывод о перспективности использования нейронных сетей типа NARX для идентификации рассматриваемой динамической системы.

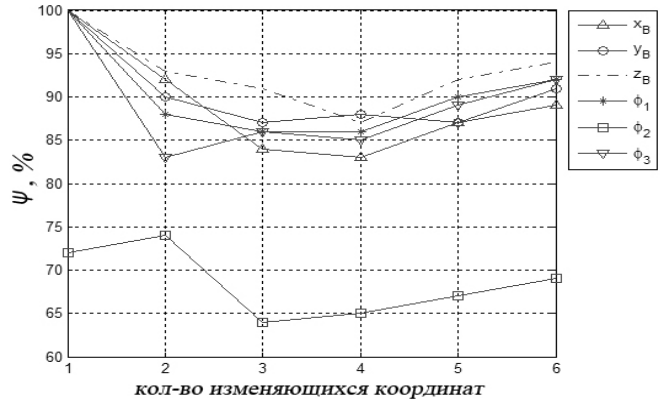


Рисунок 3 – К оценке точности нейросетевой аппроксимации гексапода: величина ψ

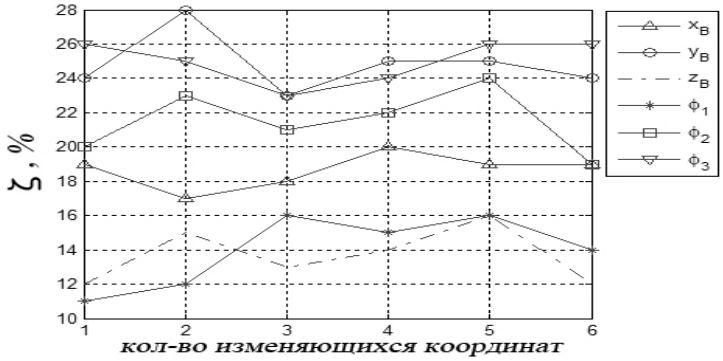


Рисунок 4 - К оценке точности нейросетевой аппроксимации гексапода: величина ζ

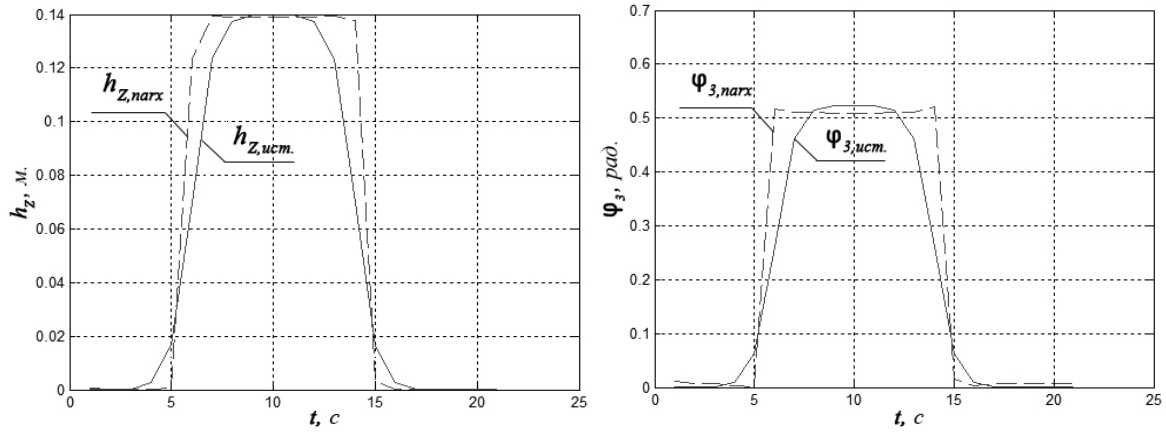


Рисунок 5 - Результаты моделирования с помощью нейронной сети обобщенных координат h_z, φ_3

Литература

1. Хайкин, С. Нейронные сети: Полный курс, 2-е издание – Пер. с англ. – М.: Изд. дом «Вильямс», 2006.
2. Разработка научных основ построения мехатронных технологических машин на базе многосекционных манипуляторов типа «хобот»; отчет о НИР (1-й этап) / МГТУ им. Н.Э. Баумана; рук. Карпенко А.П. -М., 2009. –147 с., №ГР 01201000171, Инв.№ 02201000234.

МОДЕЛИРОВАНИЕ ДИНАМИЧЕСКИХ СИСТЕМ МЕТОДОМ МОЛЕКУЛЯРНОЙ ДИНАМИКИ

Хихин Р. Р.
Научный руководитель: кандидат технических наук, доцент Назаров А.В.
Московский Государственный Технический Университет им. Н. Э. Баумана, ИУ4

MODELLING OF THE DYNAMIC SYSTEMS WITH THE USE OF MOLECULAR DYNAMIC METHOD

Hihin R.R.
Supervisor: Ph.D., Associate Professor Nazarov A.V.
Bauman Moscow State Technical University, Moscow, Russia

Аннотация
Данная работа посвящена моделированию поведения динамических систем методом молекулярной динамики. В данной работе рассмотрены физические основы метода молекулярной динамики и его реализация при помощи алгоритма Верле. Представлены результаты моделирования поведения системы из 12 частиц, на основе которого показана возможность связи параметров отдельных частиц с параметрами макроскопического тела в целом.

Annotation
This work deals with the problem of the modeling of the dynamic systems with the use of the molecular dynamic method. Physical basis of the molecular dynamic method and its realization with the use of Verle method are given. Results of the 12 atom system modeling are given. The correlation between the parameters of the particle and macroscopic body is shown.

Введение

Многие системы, такие как газы, жидкости и твердые тела, состоят из большого числа взаимодействующих друг с другом частиц. Макроскопические свойства данных тел всецело зависят от свойств составляющих их частиц. Но возникает проблема: Как можно, исходя из известных межмолекулярных взаимодействий, понять наблюдаемое поведение сложной много частичной системы. Самый очевидный ход – решить задачу в лоб, моделируя на компьютере саму систему многих частиц. Этот подход, называется методом молекулярной динамики[1].

1. Физические основы метода молекулярной динамики

Поскольку необходимо изучить качественные свойства систем многих частиц, пойдем на упрощение задачи, предполагая, что динамику можно считать классической, а молекулы – химически инертными шариками, а сила взаимодействия двух молекул зависит только от расстояния меду ними. . Наиболее важной особенностью межмолекулярного взаимодействия жидкостей является сильное отталкивание для малых расстояниях и слабое притяжение на больших расстояниях. Энергия такого межмолекулярного взаимодействия описывается при помощи потенциал Леннарда-Джонса (рисунок 1) [1]

$$V(r) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] \quad (1)$$

где $V(r)$ – энергия межмолекулярного взаимодействия;

ε – параметр, характеризующий энергию связи;

σ - параметр, характеризующий длинну связи;

r – рассояние между частицами.

Здесь и далее, энергия частиц выражается в единицах σ, масса - в единицах m, где m – масса частиц. Тогда скорость выражается в единицах $\tau = (\varepsilon/m)^{1/2}$, а время – в единицах $v = (m\sigma^2\varepsilon)^{1/2}$ [1]

На рисунке 1 представлен график зависимости потенциала ленарда-джонса в относительных единицах, где энергия выражена в единицах ε, а расстояние в единицах σ.

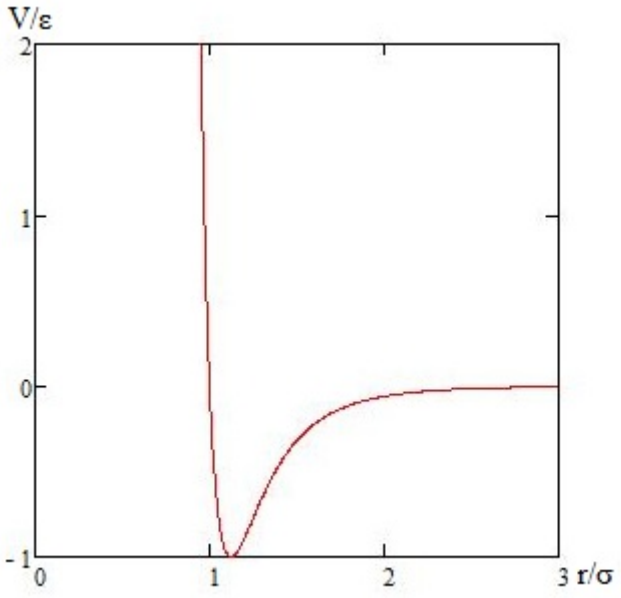


Рисунок 1 - Потенциал Леннарда-Джонса в относительных единицах

Заметим, что данный потенциал является «короткодействующим» и $V(r)$, и на больших расстояниях потенциал по существу равен нулю.

Для моделирования траекторий частиц используется алгоритм Верле [1]. Данный алгоритм основан на законах классической механики и методе конечных элеметов.

В соответствии с данным алгоритмом сила взаимодействия между двумя частицами F определяется по формуле:

$$F = -grad(V(r)) \quad (2)$$

где F – сила взаимодействия частиц;

$V(r)$ – энергия взаимодействия между частицами.

По третьему закону ньютона эта сила действует на каждую из двух взаимодействующих частиц, вызывая ускорение.

$$a = \frac{F}{m} \quad (3)$$

где F – сила взаимодействия частиц;

m – масса частицы;

Так как в рассматриваемой задаче одновременно взаимодействует набор частиц, то суммарное ускорение частицы находим как сумму ускорений, вызванных двухчастичным взаимодействием [1].

Далее после нахождения нового ускорения находится новая скорость частицы v_{n+1}

$$v_{n+1} = v_n + \frac{a_n + a_{n+1}}{2}\Delta t \quad (4)$$

где v_{n+1} - значение скорости на шаге (n+1);

v_n – значение скорости на шаге n;

a_{n+1} – значение ускорения на на шаге (n+1);

a_n – значение ускорения на на шаге n;

Δt – изменение времени за шаг.

Далее находится новая координата частицы x_{n+1}

$$x_{n+1} = x_n + v_n \Delta t + \frac{a_1 \Delta t^2}{2} \quad (5)$$

где x_{n+1} – координата частицы на шаге (n+1);

x_n – координата частицы на n шаге n;

v_n – скорость на n-ом шаге

a_n – ускорение на шаге n;

Δt – изменение времени за шаг.

Далее новая координата используется для нахождения нового ускорения a_{n+1}, которое вместе с ускорением a_n используется для получения новой скорости v_{n+1}.

2. Краевые условия

В моделях молекулярный динамики моделируется несколько сотен частиц, в то время как макроскопическое тело состоит из 10^{23} частиц, поэтому доля частиц вблизи стенок не мала, и возникают различные поверхностные эффекты. Для того что бы перенести результаты моделирования малого количества частиц на макроскопическую систему состоящую из 10^{23} частиц необходимо ввести специальные граничные условия [1].

Наиболее просто и точно промоделировать свойства макроскопической системы можно используя периодические краевые условиях, которые заключаются в том что частица, попадая в стенку, не отражается от нее, а входит в неё, выходя из противоположной стенки [1].

3. Практическая реализация модели

На основе метода молекулярной динамики была промоделирована система состоящая из N=12 частиц, для чего была составлена соответствующая программа на языке Delphi. Изначально все частицы удерживались вместе в левой верхней части экрана. Затем они отпускались и начинали взаимодействовать между собой.

Изначально удерживаемые частицы в левой половине ящика, при разрушении удерживающих их связей начинают разлетаться.

$t = 0\,\tau$ $t = 0.657\tau$ $t = 1.14\tau$

Рисунок 2 – Поведение системы состоящей из N=12 частиц с течением времени

Как и следовало того ожидать частицы стремятся к равновесию. Качественным критерием, по которому мы можем судить о том что система находится в равновесии, является выравнивание концентрации по всему объему.

Для простоты разделим «ящик» на 2 области – левую и правую, и проследим изменение количества частиц в данных областях. На рисунке 3 представлен график изменения.

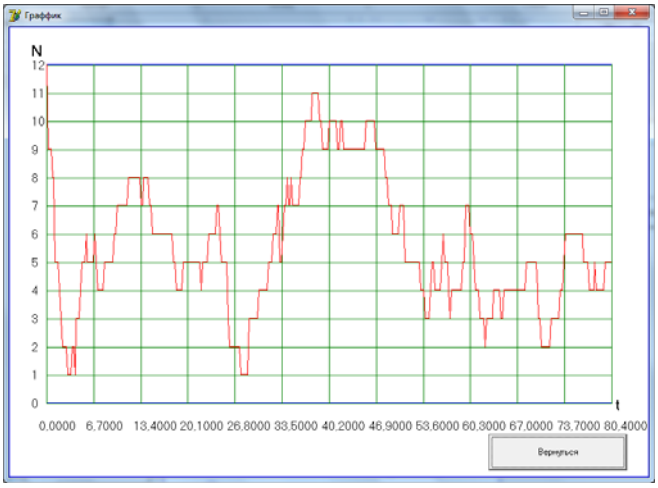


Рисунок 3 – График изменения количество частиц в левой половине от времени.

Как видно по графику изначально количество частиц слева максимально, потом оно резко снижается и со временем (приблизительно при t=60 τ) начинает колебаться относительно среднего значения. Значительная амплитуда колебания объясняется малым общим количеством промоделированных частиц.

На основе полученных данных о траекториях полета частиц можно путем интегрирования не только оценить не только концентрации частиц, но и другие параметры, характеризующие не только отдельные частицы, но и макроскопическое тело в целом, такие как температура тела, давление энтропия.

Например зная скорости частиц и энергию их межмолекулярного потенциального взаимодействия можно определить температуру макроскопического тела.

Таким образом используя метод молекулярной динамики можно оценить параметры макроскопического тела, моделируя поведение ограниченного количества отдельно составляющих его частиц, и распространения результата на всё макроскопическое тело.

Литература

1. Х.Гулд, Я.Тобочник "Компьютерное моделирование в физике" Мир, 1990 т.1

ПОРТАЛ-ХАБ ННС

Верстов В.А.
Научный руководитель: кандидат технических наук, доцент Власов А. И.
МГТУ им. Н. Э. Баумана, г. Москва, Россия

PORTAL HUB NNS

Verstov V.A.
Supervisor: Ph.D., Associate Professor Vlasov A. I.
Bauman Moscow State Technical University, Moscow, Russia

Аннотация
В статье рассматривается «Портал-хаб ННС» для высокопроизводительных вычислений в области инженерного анализа и проектирования в наноиндустрии. Рассмотрена его архитектура и программное обеспечение. Приведен пример использования вычислительного кластера для моделирования наносистем. Описан информационный портал вычислительного кластера. Разработаны образовательные программы использующие ресурсы «Портала-хаба ННС».

Abstract
“Portal Hub NNS” for high-perfomance computing in engineering analysis and design in the nanosystems are described in the article. The architecture and software that used in this system are also described. It was shown the opportunity of using computing cluster for the modeling nanosystems. There is some information about website of computing cluster in this article. It was suggested educational programs that used the resoures of “Portal Hub NNS”.

Введение

В настоящее время задачи квантовой механики, молекулярной динамики, фотоники требуют значительных вычислительных мощностей и использования параллельных вычислений. Есть два подхода к решению проблемы получения достаточных вычислительных мощностей для указанных задач: использование суперкомпьютерных или кластерных систем.

В МГТУ им. Н.Э. Баумана есть своя инфраструктура высокопроизводильтельных вычислений:

- Основной кластер МГТУ (http://iu9.bmstu.ru/science/clusters_discovery.php) - проводятся исследования производительности кластерных установок, а так же решаются прикладные задачи для высокопроизводительных компьютеров
- Центр компетенции IBM по мейнфреймам (http://mainframe.bmstu.ru/) - академический центр компетенции IBM в области больших вычислительных машин
- Вычислительный кластер ННС (http://cluster.iu4.bmstu.ru/) – входит в состав «Портала-хаба ННС»

Классификация кластерных систем

Существует два подхода к реализации высокопроизводительных вычислений: многопроцессорные вычислительные системы и многокомпьютерные вычислительные системы. Причем каждый отдельный вычислительный узел такой системы может представлять собой многопроцессорную систему. Существует несколько видов возможной архитектуры таких компьютеров[3]:

1. Uniform Memory Access (сокращённо UMA — «однородный доступ к памяти») — архитектура многопроцессорных компьютеров с общей памятью. Все микропроцессоры в

UMA-архитектуре используют физическую память одновременно. При этом время запроса к данным из памяти не зависит ни от того, какой именно процессор обращается к памяти, ни от того, какой именно чип памяти содержит нужные данные. Однако каждый микропроцессор может использовать свой собственный кэш.

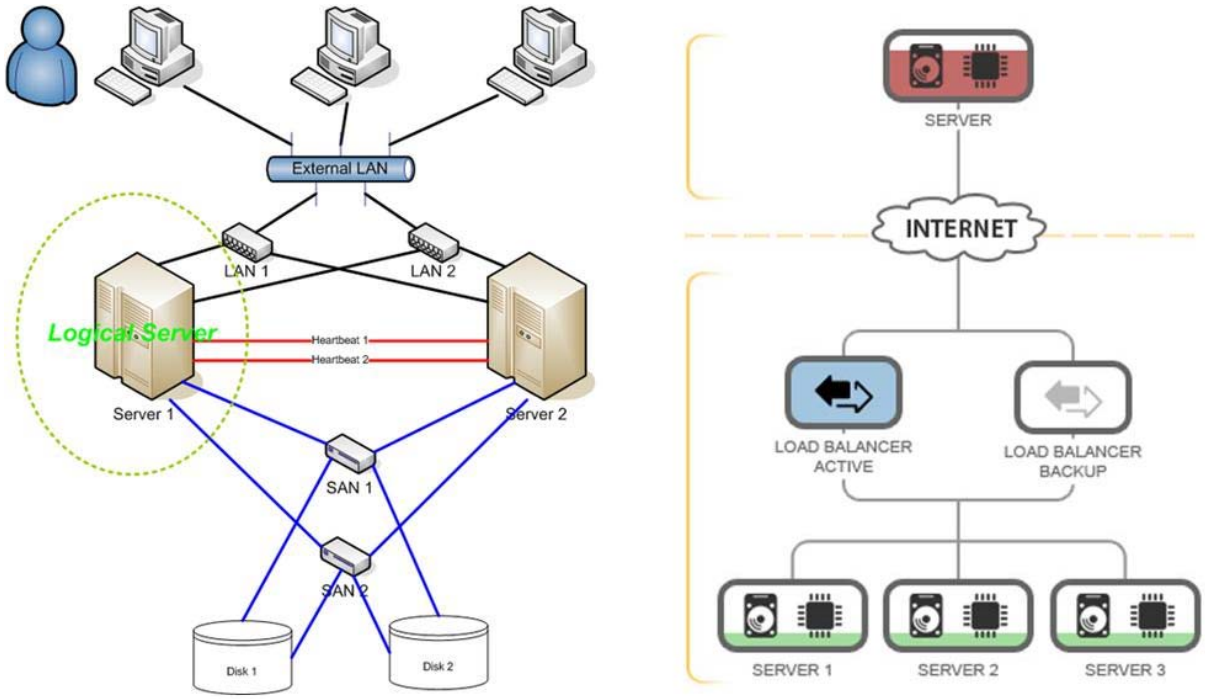
2. NUMA (Non-Uniform Memory Access — «неравномерный доступ к памяти» или Non-Uniform Memory Architecture — «Архитектура с неравномерной памятью») — схема реализации компьютерной памяти, используемая в микропроцессорах, когда время доступа к памяти определяется её расположением по отношению к процессору.

Мультикомпьютер — вычислительная система без общей памяти, состоящая из большого числа взаимосвязанных компьютеров, у каждого из которых имеется собственная память. Процессоры мультикомпьютера отправляют друг другу сообщения (используется сетевая топология двух-, трехмерной решетки или деревья и кольца)[2]. Причина разработки — техническая сложность создания мультипроцессоров. Кластер – одна из наиболее эффективных реализаций мультикомпьютерных систем.

Кластер — группа компьютеров, объединённых высокоскоростными каналами связи и представляющая с точки зрения пользователя единый аппаратный ресурс.

Виды кластеров[1]:

1. Кластеры высокой доступности - обозначаются аббревиатурой HA (англ. High Availability — высокая доступность). Создаются для обеспечения высокой доступности сервиса, предоставляемого кластером.



а) б)

Рисунок 1 - Варианты реализации ресурсного центра (а) структура кластера высокой доступности, б) структура кластера распределения нагрузки)

Избыточное число узлов, входящих в кластер, гарантирует предоставление сервиса в случае отказа одного или нескольких серверов. Типичное число узлов — два, это минимальное количество, приводящее к повышению доступности. Создано множество программных решений для построения такого рода кластеров.

2. Кластеры распределения нагрузки - принцип их действия строится на распределении запросов через один или несколько входных узлов, которые перенаправляют их на обработку в остальные, вычислительные узлы.

Первоначальная цель такого кластера — производительность, однако, в них часто используются также и методы, повышающие надёжность. Подобные конструкции называются серверными фермами.

3. Вычислительные кластеры - кластеры используются в вычислительных целях, в частности в научных исследованиях. Специально выделяют высокопроизводительные кластеры (Обозначаются англ. аббревиатурой HPC Cluster — High-performance computing cluster).

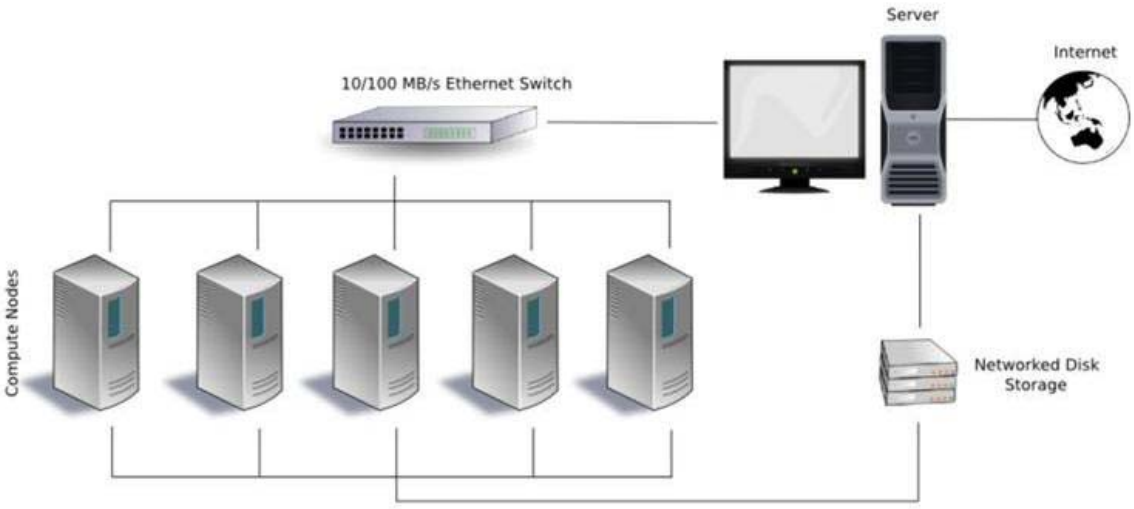


Рисунок 2 - Структура вычислительного кластера

Для вычислительных кластеров существенными показателями являются высокая производительность процессора в операциях над числами с плавающей точкой (flops) и низкая латентность объединяющей сети, и менее существенными — скорость операций ввода-вывода. Вычислительные кластеры позволяют уменьшить время расчетов, по сравнению с одиночным компьютером, разбивая задание на параллельно выполняющиеся ветки, которые обмениваются данными по связывающей сети.

«Портал-хаб ННС»

В рамках ФЦП «Кадры» НОЦ «Наносистемы» кафедры ИУ4 создали комплексный «Портал-хаб ННС» для высокопроизводительных вычислений в области инженерного анализа и проектирования в наноиндустрии. Целью для создания данного ресурса послужило создание и развитие в ННС вычислительного портала для ускоренного внедрения высокопроизводительных вычислений для инженерного анализа и проектирования на базе специализированного прикладного программного обеспечения в практику разработки оборудования в наноиндустрии, повышения эффективности работ на стадиях НИР и ОКР, для внедрения в других отраслях отечественной промышленности и высшей школе, а также создание информационного портала вычислительного кластера по направлению «Наноинженерия», предоставляющего возможности решения модельных задач объектов наноинженерии, используя специализированного прикладного ПО.

«Портал-хаб ННС» состоит из информационного портала вычислительного кластера, пользовательского ресурсного хаба и вычислительного кластера ННС.

Вычислительный кластер ННС

Реализован удаленный доступ к вычислительным ресурсам кластера по протоколу SSH. Доступ к вычислительным ресурсам предоставляется при условии соблюдения правил в соответствии с документом "Политика безопасности вычислительных ресурсов кластера" Доступ предоставляется пользователям, зарегистрированным непосредственно на кластере. Для регистрации нового пользователя необходимо заполнить анкету и отправить ее на электронный адрес cluster@iu4.ru. При подключении к кластеру необходимо следовать подробной инструкции по подключению к кластеру. Порядок подключения показан на рисунке 3.

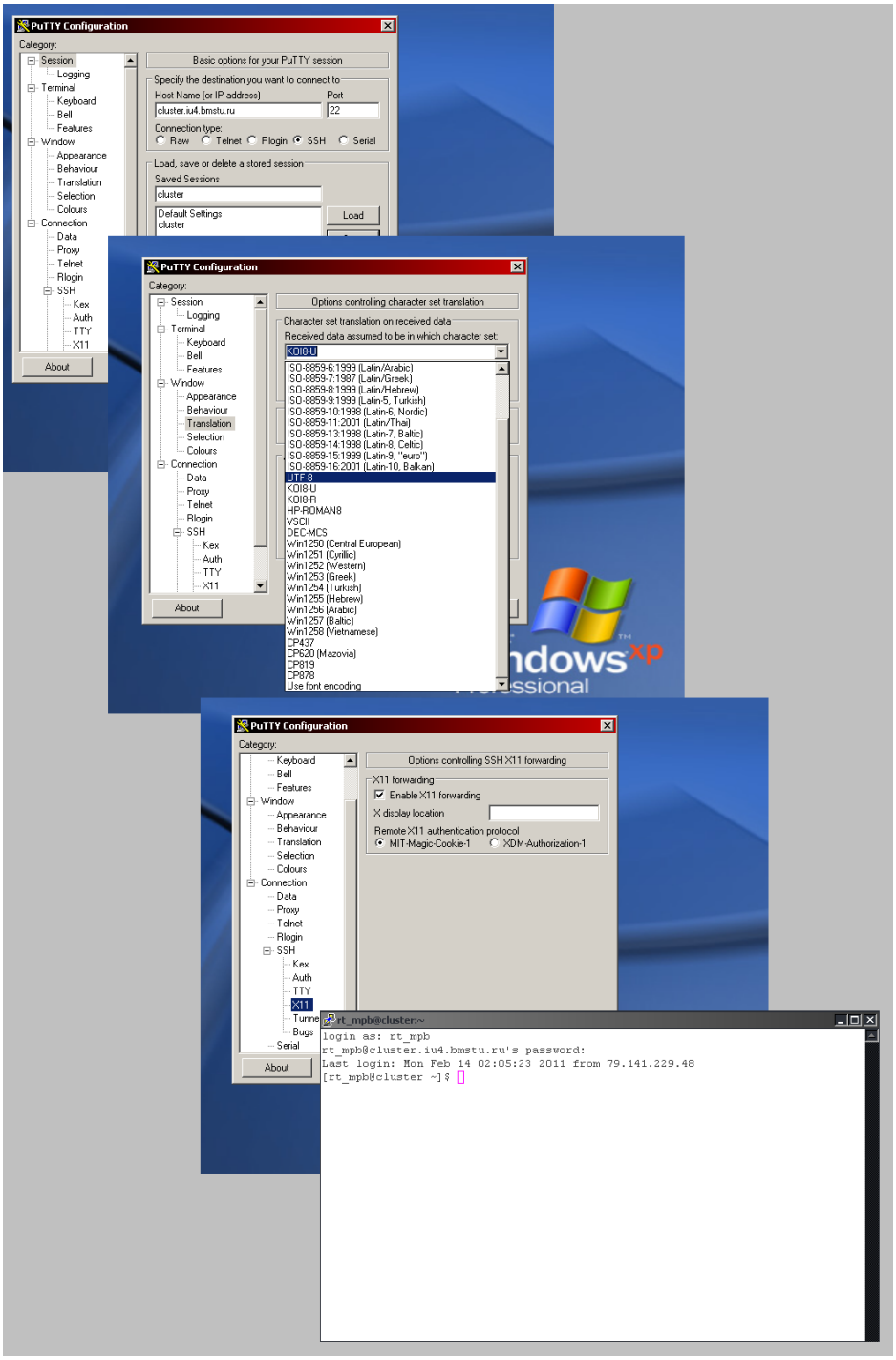


Рисунок 3 - Подключение к «Вычислительному кластеру ННС»

Кластер был развернут для решения задач проектирования микро- и наносистем, требующих высокопроизводительных вычислительных ресурсов. Для текущей работы доступна система следующей конфигурации:

Таблица 1 - Архитектура вычислительного кластера

Сервера HP	2 x HP DL360R05 (Dual Intel® Xeon® Processor E5430 (12M Cache, 2.66 GHz, 1333 MHz FSB)/16G, 75 G)
RISC сервера IBM	2 x IBM OpenPower P5 RiskServer 710
Система хранения данных	NetApp FAS2050-R5 в составе:FAS2050,NetApp Select,20x750GB SATA,-C,R5
Система резервного копирования	Sun SL48 w/ 1HP LTO4 SCSI
Коммуникационная инфраструктура	Gigabit Ethernet на основе оборудования Cisco

Таблица 2 - Общесистемное программное обеспечение вычислительного кластера

Операционная система	CentOS
Кластерное ПО	MOSIX
Интерфейс передачи сообщений между процессами	MPI2
Система мониторинга	Ganglia

Таблица 3 - Прикладное программное обеспечение кластера

MIT Photonic Bands	Свободно распространяемый пакет программ для расчета электромагнитных волн в периодических диэлектриках. Это может быть использовано для решения различных инженерных задач в оптике, например проектирование фотонных кристаллов, волноводов и резонаторных систем.
NWChem	Пакет позволяет исследовать вопросы химических процессов с применением теоретических методов для прогнозирования: структуры и свойств.
Scilab	Пакет прикладных математических программ, предоставляющий мощное открытое окружение для инженерных (технических) и научных расчётов. Scilab имеет схожий с MATLAB язык программирования. В состав пакета входит утилита, позволяющая конвертировать документы Matlab в Scilab.
Прикладное ПО собственной разработки	Вычислительные комплексы по фотонике и опто-электронике, САПР по технологии двойного фотошаблона

В качестве примера эффективности использования параллельных вычислений приведем график (рис. 4) зависимости времени расчета электромагнитного поля в алмазоподобной кристаллической решетке в пакете MIT Photonic Bands (MPB), который поддерживает параллельные вычисления с помощью программного интерфейса Message Passing Interface (MPI, интерфейс передачи сообщений) на кластере MOSIX. Расчет выполнялся на «Вычислительном кластере ННС», работающем на кафедре ИУ-4 МГТУ им. Н.Э. Баумана.

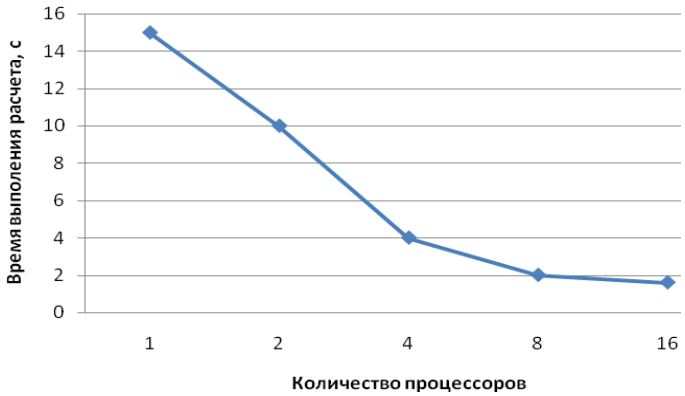


Рисунок 4 - Зависимость времени расчета электромагнитного поля в алмазоподобной кристаллической решетке в пакете MPB от числа процессоров.

Результат и порядок выполнения расчета электромагнитных полей в периодических диэлектриках показан на рисунке 5.

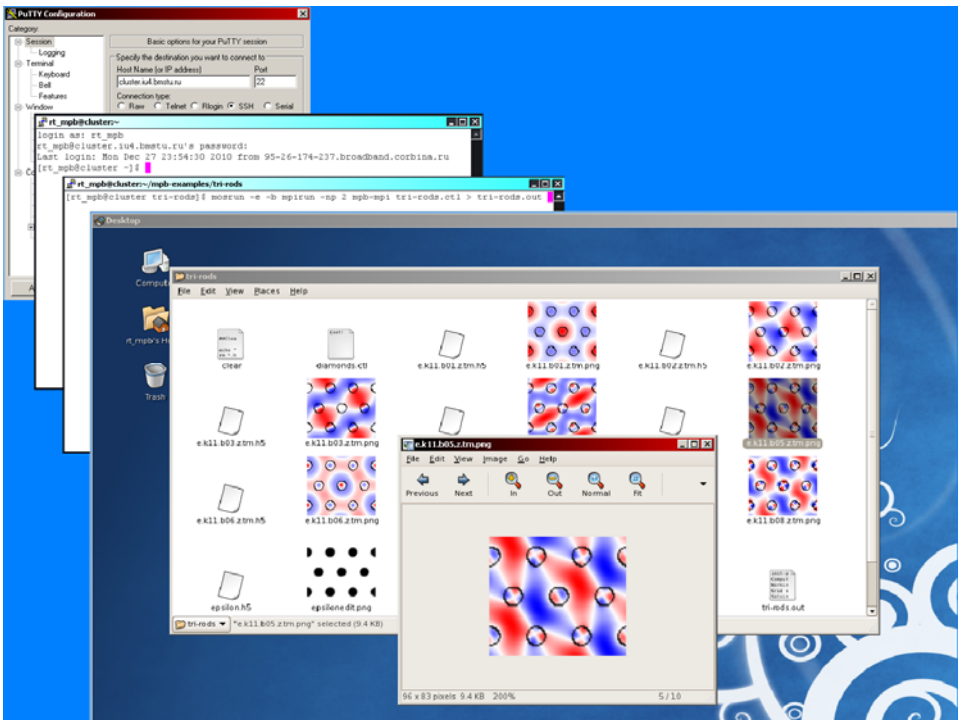


Рисунок 5 - Использование кластера для расчета электромагнитного поля в периодических диэлектриках при помощи пакета MPB

Для проведения моделирования на кластере доступны следующие пакеты:

MIT Photonic Bands – свободно распространяемый пакет программ для расчета электромагнитных волн в периодических диэлектриках. MPB был разработан Стивеном Джонсоном из Массачусетского технологического института.

MPI — программный интерфейс (API) для передачи информации, который позволяет обмениваться сообщениями между процессами, выполняющими одну задачу. Разработан Уильямом Гроуппом, Эвином Ласком и другими.

MPI является наиболее распространённым стандартом интерфейса обмена данными в параллельном программировании, существуют его реализации для большого числа компьютерных платформ. Используется при разработке программ для кластеров и суперкомпьютеров. Основным средством коммуникации между процессами в MPI является передача сообщений друг другу. Существуют реализации для языков Фортран 77/90, Си и Си++. В первую очередь MPI ориентирован на системы с распределенной памятью, то есть когда затраты на передачу данных велики[4].

MOSIX это система управления кластерами и сетями ОС на ядре Linux, представляющая их как одну систему (Single-System Image, SSI), то есть эквивалент операционной системы для кластера в целом. В кластере MOSIX нет необходимости в модификации существующих приложений, в связывании с дополнительными библиотеками, в явном входе на удаленные узлы — все это осуществляется автоматически, прозрачно для приложений подобно SMP[1].

Пользовательский ресурсный хаб

Пользовательский ресурсный хаб host.iu4.bmstu.ru, предоставляет студентам возможность размещения своих информационных и расчетных проектов, доступ к хранилищам данных, программным библиотекам, сервисам (FTP, WWW, PHP, MySQL, Oracle и т.п.).

Для каждого пользователя создается домен http://host.iu4.bmstu.ru/~fio. Каждый студент, зарегистрированный в системе, имеет возможность хранить свои материалы на своем хостинге, учебные материалы по курсам «Основы конструкторско-технологической информатики», «Системное программирование», «САПР наносистем», «Конструкторско-технологические базы данных».

Сервер host.iu4.bmstu.ru является управляющим для систем хранения и кафедральным архивом.

Программное решение такой задачи - программный комплекс Webmin.

Webmin - это программный комплекс, позволяющий администрировать операционную систему через веб-интерфейс, в большинстве случаев, позволяя обойтись без использования командной строки и запоминания системных команд и их параметров. Используя любой браузер, администратор сервера может создавать новые учётные записи пользователей, почтовые ящики, изменять настройки служб и сервисов, например: веб-сервера Apache, DNS. Однако, в некоторых случаях необходимо знание операционной системы и редактирование конфигурационных файлов вручную. Кроме того, не все возможности операционной системы и не все программы можно конфигурировать через интерфейс Webmin, например nginx пока не входит в базовый набор.

Webmin состоит из простого веб-сервера и большого количества скриптов (>500), которые собственно и осуществляют связь между командами администратора через веб-интерфейс и их исполнением на уровне операционной системы и прикладных программ. Webmin написан полностью на языке Perl и не использует никаких дополнительных нестандартных модулей. Простота, лёгкость и быстрота выполнения команд - одно из самых больших преимуществ данной панели управления.

Другое важное преимущество - возможность исправлять конфигурационные файлы вручную, так как Webmin не "портит" конфигурационные файлы, в отличие от некоторых других панелей управления, и следует, как правило, политикам дистрибутивов по конфигурированию программ.

Данная панель управления бесплатно распространяется для коммерческого и некоммерческого использования. Авторы этой программы позволяют всем желающим не только бесплатно использовать программу, но и изменять её по своему усмотрению.

Работать с Webmin достаточно просто - нужно запустить браузер, набирать https://host.iu4.bmstu.ru:10000, ввести свою авторизационную информацию (логин и пароль) и попасть на страницу администрирования.

При использовании системы Webmin необходимо следовать подробной инструкции пользователя.

Информационный портал вычислительного кластера

Информационный портал вычислительного кластера доступен по адресу http://cluster.iu4.bmstu.ru. Ресурс посвящен освещению работы вычислительного кластера по направлению «Наноинженерия». На данном портале размещена информация по учебным курсам для школьников, бакалавров, магистров и аспирантов по направлению "Проектирование и технология производства ЭС" (кафедра ИУ4 МГТУ им. Н.Э. Баумана).

Программа для школьников предназначена для учащихся 10-11 классов (возможно и более младших, отбор по собеседованию) общеобразовательных средних школ естественно - научного, физико-математического и подобных профилей. Учащимся профильных физико-математических школ предоставляется возможность познакомиться с основами высокопроизводительных вычислений в области нанотехнологий.

В рамках подготовки специалистов и бакалавров/магистров реализуются образовательные программы подготовки специалистов в области высокопроизводительных кластерных вычислений в наноинженерии. Программа включает в себя курс "Основы конструкторско-технологической информатики".

Магистерские программы ориентированы на изучение системного программирования, проведение исследований разнообразных электромагнитных волн, химических процессов в наносистемах, моделирования и проектирование различных инженерных задач в оптике, например проектирование фотонных кристаллов, волноводов и резонаторных систем.

В распоряжение аспирантов предоставляется доступ к вычислительным ресурсам кластера, у них появляется возможность моделировать и проводить расчеты для своих проектов с помощью установленного на кластере программного обеспечения, а так же участвовать в разработке прикладного ПО.

Литература

1. Barak A., Guday S. and Wheeler R., The MOSIX Distributed Operating System, Load Balancing for UNIX. Lecture Notes in Computer Science, Vol. 672, Springer-Verlag, 1993, с. 37, с. 136-138
2. Суперкомпьютерные технологии в науке, образовании и промышленности/Под редакцией: академика В. А. Садовничего, академика Г. И. Савина, чл.-корр. РАН Вл. В. Воеводина.-М.: Издательство Московского университета, 2009.-232 с.
3. Rodgers, David P. (June 1985). "Improvements in multiprocessor system design". ACM SIGARCH Computer Architecture News archive (New Yorok, NY, USA: ACM) 13 (3): 225–231 с.
4. Using MPI: Portable Parallel Programming with the Message-Passing Interfaceby William Gropp, Ewing Lusk, and Anthony Skjellum, 1994, с. 124

Работа выполнена в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009 – 2013 годы

ПРИМЕНЕНИЕ НЕЧЕТКОЙ ЛОГИКИ ДЛЯ ПРОЕКТИРОВАНИЯ КОРПУСОВ ЭЛЕКТРОННЫХ СРЕДСТВ

Михеев Г.А.
Научный руководитель: к.т.н., доцент Журавлева Л.В.
Московский государственный технический университет им. Н.Э.Баумана, Москва, Россия

APPLICATION OF FUZZY LOGIC FOR DESIGNING CASES FOR ELECTRONIC EQUIPMENT

Mikheev G.A.
Scientific adviser: Ph. D., docent Zguravleva L.V.
Moscow State Technical University n.a. N.E. Bauman, Moscow, Russia

Аннотация
В работе рассматривается проблема поиска оптимального конструкторского решения с применением теории нечетких множеств в процессе проектирования корпусов электронной аппаратуры.

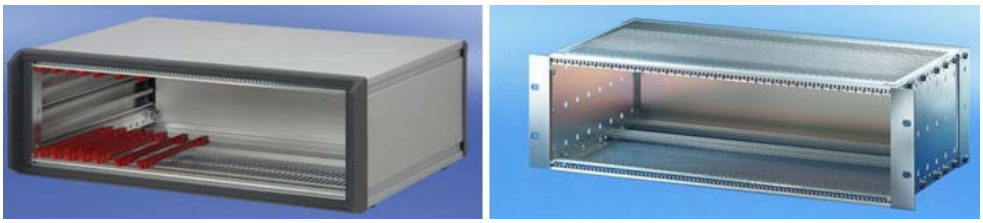
Abstract
This paper considers the problem of finding the optimal design solutions with help of the theory of fuzzy sets during the design cases for electronic equipment.

Для защиты электронных средств от воздействия окружающей среды, таких дестабилизирующих факторов как вибрация, механические повреждения, пыль, влага, радиация, перепады температур используют корпуса. Конструкции корпусов зависят от требований технических заданий на проектирование электронных средств и характеризуются большим разнообразием.

Способность корпуса выполнять свои функции оценивается показателями качества. Номенклатура показателей качества, наиболее полно характеризующая качество корпусов электронных устройств, представлена на рис. 1.

Рисунок 1 - Показатели качества корпусов электронной аппаратуры

На рисунке 1 показаны 19-дюймовые Конструкции корпусов для установки на стол и в шкаф представлены на рис.2.



а *б*

Рисунок 2 – 19-дюймовые корпуса; *а* - настольный, *б* – для установки в шкаф

Конструкции корпусов электронных средств для установки на стол и в шкаф представлены на рис.2, для крепления на стену - на рис.3.

Рисунок 3 – Конструкция корпуса для крепления устройства на стену

При конструировании корпусов электронной аппаратуры в техническом задании могут быть оговорены требования, которые нельзя оценить численно. Примерами таких требований могут быть «эстетичность», «удобство эксплуатации», «простота установки», «патентная чистота», «ремонтопригодность» и т.п. В этом случае возникает неопределенность и многокритериальность поставленных задач. При поиске рационального проектно-конструкторского решения неопределенность, присущую рассматриваемой нечетко определенной конструкторской задаче (НПКЗ), рекомендуется сохранять, чтобы обеспечить полноту состава исходной информации.

Для того чтобы свести НПКЗ к формально разрешимым задачам надо тем или иным образом «снять неопределенности», т. е. либо ввести гипотезы, либо назначить оценки. Следовательно, при решении НПКЗ не обойтись без помощи экспертов, способных формально описать нечетко определенную проблемную ситуацию на языке, понятном электронно- вычислительной машине (ЭВМ) [1].

В целях формализации процедур, связанных с неколичественными измерениями, используются специальные функции принадлежности и на их основе вводятся лингвистические переменные, которые в наиболее естественной для человека-конструктора форме отражают особенности его неформальных операций и в то же время являются точными операндами для ЭВМ.

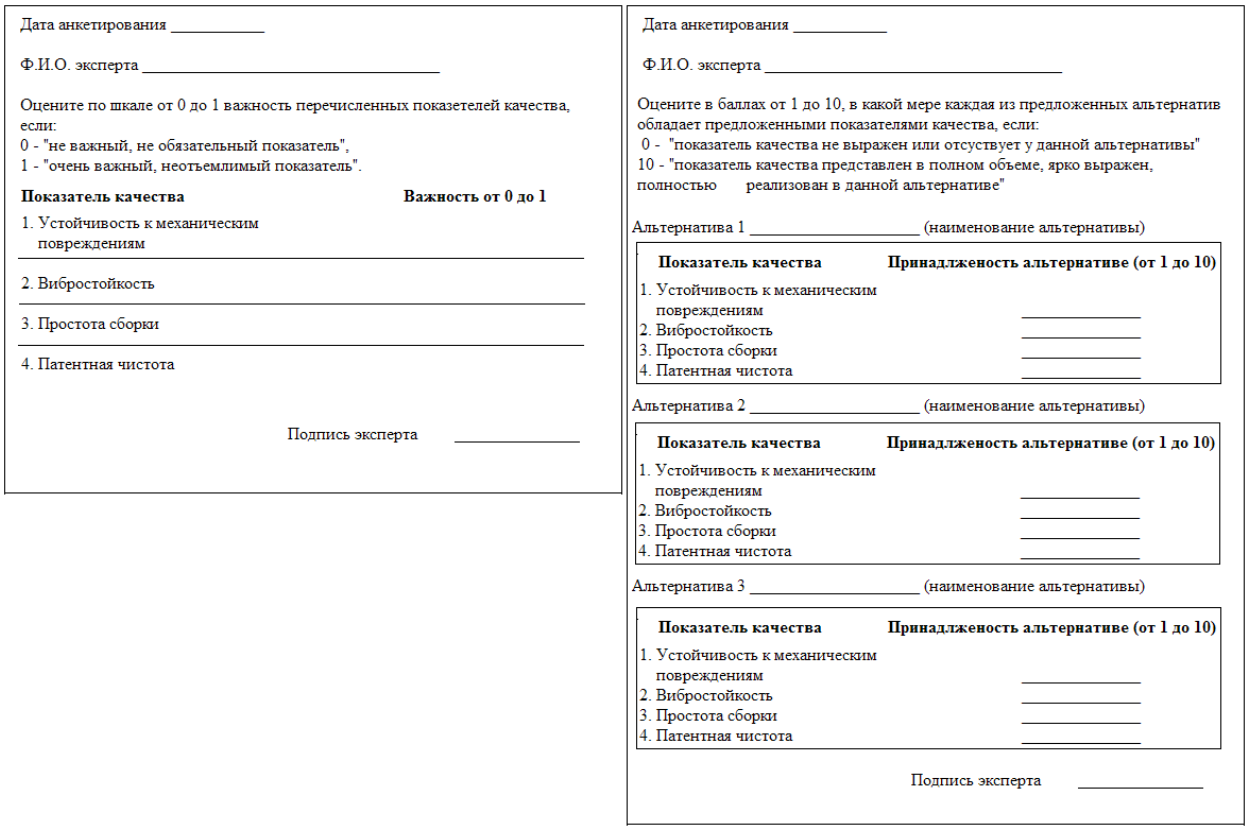
Методика получения значений качественных характеристик рассматриваемых проектно-конструкторских решений основывается на принципах экспертного оценивания. Опрос экспертов представляет собой заслушивание и фиксацию в содержательной и

количественной форме суждений экспертов по решаемой проблеме. Основными видами опроса экспертов являются: анкетирование и интервьюирование; дискуссии; метод «мозгового штурма»; метод Дельфы; метод «суда».

В условиях ограниченных временных и прочих ресурсов, наиболее подходящим является вид метода индивидуального опроса – анкетирование.

Анкетирование представляет собой опрос экспертов с помощью анкет, на вопросы которых они должны дать ответы в письменной форме, либо с использованием технических средств.

В ходе данной работы экспертам были предложены две анкеты. Первая анкета ставит целью оценить степень важности показателей качества, присущих каждой из альтернатив. Под альтернативой понимается конкретный вариант конструкции корпуса устройства, предлагаемый исполнителем заказчику. Вторая анкета позволяет получить конкретные значения функций принадлежности, то есть определяет, насколько сильно выражен конкретный показатель качества для каждой альтернативной конструкции корпуса. Формы анкет представлены на рисунке 4.



Дата анкетирования _____

Ф.И.О. эксперта _____

Оцените по шкале от 0 до 1 важность перечисленных показетелей качества, если:
0 - "не важный, не обязательный показатель",
1 - "очень важный, неотъемлимый показатель".

Показатель качества	Важность от 0 до 1
1. Устойчивость к механическим повреждениям	
2. Вибростойкость	
3. Простота сборки	
4. Патентная чистота	

Подпись эксперта _____

Дата анкетирования _____

Ф.И.О. эксперта _____

Оцените в баллах от 1 до 10, в какой мере каждая из предложенных альтернатив обладает предложенными показателями качества, если:
0 - "показатель качества не выражен или отсутствует у данной альтернативы"
10 - "показатель качества представлен в полном объеме, ярко выражен, полностью реализован в данной альтернативе"

Альтернатива 1 _____ (наименование альтернативы)

Показатель качества	Принадлженость альтернативе (от 1 до 10)
1. Устойчивость к механическим повреждениям	_____
2. Вибростойкость	_____
3. Простота сборки	_____
4. Патентная чистота	_____

Альтернатива 2 _____ (наименование альтернативы)

Показатель качества	Принадлженость альтернативе (от 1 до 10)
1. Устойчивость к механическим повреждениям	_____
2. Вибростойкость	_____
3. Простота сборки	_____
4. Патентная чистота	_____

Альтернатива 3 _____ (наименование альтернативы)

Показатель качества	Принадлженость альтернативе (от 1 до 10)
1. Устойчивость к механическим повреждениям	_____
2. Вибростойкость	_____
3. Простота сборки	_____
4. Патентная чистота	_____

Подпись эксперта _____

Рисунок 4 – Формы анкет для опроса экспертов

В качестве экспертов привлекались опытные специалисты ЗАО МЦСТ, которые занимаются разработкой корпусов для средств вычислительной техники частного применения.

Обработка экспертных оценок. После проведения опроса группы экспертов была проведена обработка результатов с целью получения обобщенных данных и новой информации, содержащейся в скрытой форме в экспертных оценках.

Для упорядочения нечетких чисел существует множество методов, которые отличаются друг от друга способом свертки и построения нечетких отношений.

Для множества альтернатив $A_i = \{a_1 \ldots a_n\}$ каждая альтернатива характеризуется нечетким множеством $A_i=\{\mu_{Ai}(x)/x\}$, полученным в результате опроса экспертов, $x \in R$ где R – множество критериев выбора.

Множество оптимальных альтернатив $B=\{\mu_b(i)/i\}$, где $\mu_b(i)$ может рассматриваться как степень соответствия альтернативы a_i понятию «наилучшая альтернатива». Для этого вводится отношение:

$$P_{ij}=\{\ \mu_{Pij}(x_i, x_j)/(x_i, x_j)\}, \qquad (1)$$

где $\mu_{Pij}(x_i, x_j)$ выражает степень превосходства x_i над x_j.

Различие в полезности значения x_i, x_j, представляется в виде:

$$\mu_{Pij}=f(x_i, x_j) = u(x_i)-u(x_j) \qquad (2)$$

Множество оптимальных альтернатив определяется как пересечение декартов произведения нечетких оценок, задающих альтернативы и отношения предпочтения P:

$$P_{ij} \cap (A_i \times A_j) \qquad (3)$$

Степень принадлежности альтернативы a_i множеству B находится как максимальное значение функции принадлежности:

$$\mu_b(i) = \sup_{\min} (\mu_{Ai}(x_i); \mu_{Aj}(x_j); \mu_{Pij}(x_i, x_j)), \qquad (4)$$

где $\sup_{\min}$ обозначает операцию максимальной свертки.

При выборе конструкции корпуса для электронной ячейки, наиболее полно соответствующей требованиям технического задания, в качестве альтернатив были предложены три различные конструкции корпусов (см. рис. 1,2). Для оценки альтернатив применялись следующие критерии (см. рис.4, анкета 2):

F1 – устойчивость к механическим повреждениям;

F2 – вибростойкость;

F3 – простота сборки;

F4 – патентная чистота.

На первом этапе были построены функций принадлежности, соответствующие понятиям, выраженным в условных единицах измерений, нечетко. Построение этих функций проводилось экспертами, располагающими знаниями не только о параметрах выбираемых альтернатив, а в гораздо более широком диапазоне. Для решения задачи выбора конструкции корпуса были использованы уже разработанные функции принадлежности, заранее составленные экспертами (рис.5) [2].

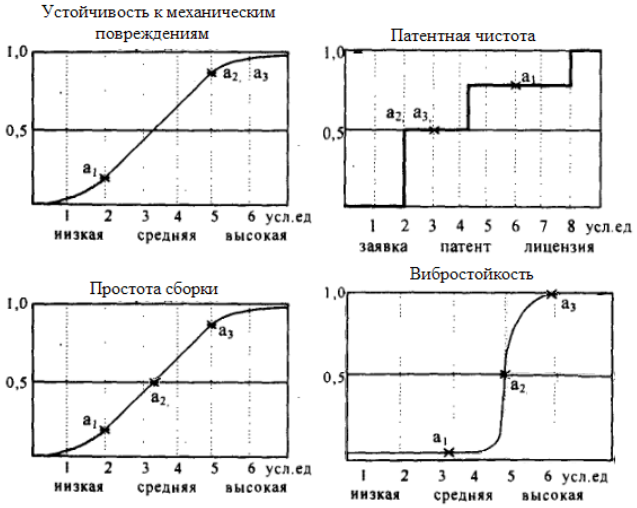


Рисунок 5 – Графики функций принадлежности для различных параметров

На втором этапе были определены конкретные значения функций принадлежности по выбранным критериям для каждого из трех альтернативных вариантов конструкций корпусов. На рисунке значения функций принадлежности, соответствующие названным альтернативам, отмечены звездочками.

Нечеткие множества для рассматриваемых критериев имеют вид:

$\mu_{F1} = 0.2/2 + 0.8/5 + 0.8/5$

$\mu_{F2} = 0.1/3 + 0.5/5 + 1/6$

$\mu_{F3} = 0.2/3 + 0.5/3.5 + 0.7/5$

$\mu_{F4} = 0.5/3 + 0.5/3 + 0.75/6$

На третьем этапе были получены весовые коэффициенты α_{ij} критериев каждой из альтернатив. Для этого варианты ответов экспертов были сведены в матрицу A, строки которой представляют собой множество показателей качества каждой альтернативы конструкции корпуса, столбцы – множество экспертов, которые проводили оценку.

$$\mathbf{A} = \|a_{tf}\|_{4\times 5} = \begin{Vmatrix} 1 & 2 & 3.5 & 3 & 4 \\ 2.5 & 2 & 1.5 & 2 & 1 \\ 2.5 & 2 & 3 & 1 & 2 \\ 4 & 3.5 & 3.5 & 4 & 1 \end{Vmatrix}$$ – матрица оценок объектов экспертами

Расчет коэффициентов компетентности экспертов и обобщенной оценки объектов проводился по формулам (5,6):

$$\mathbf{B}\vec{x} = \lambda_{\mathbf{B}}\vec{x}, \quad \sum_{t=1}^{n} x_t = 1, \quad \mathbf{C}\vec{k} = \lambda_{\mathbf{C}}\vec{k}, \quad \sum_{f=1}^{m} k_f = 1, \qquad (5, 6)$$

где матрицы $\mathbf{B} = \mathbf{A}\mathbf{A}^{\mathrm{T}}, \mathbf{C} = \mathbf{A}^{\mathrm{T}}\mathbf{A}$, $\vec{x}, \vec{k}$ — собственные векторы матриц B и C, соответствующие максимальным собственным числам этих матриц $\lambda_{\mathbf{B}}$, $\lambda_{\mathbf{C}}$.

Расчет матриц B и C:

$$\mathbf{B} = \mathbf{A}\mathbf{A}^{\mathrm{T}} = \begin{Vmatrix} 42.25 & 21.75 & 28.00 & 39.25 \\ 21.75 & 17.50 & 18.75 & 31.25 \\ 28.00 & 18.75 & 24.25 & 33.50 \\ 39.25 & 31.25 & 33.50 & 57.50 \end{Vmatrix}, \quad \mathbf{C} = \mathbf{A}^{\mathrm{T}}\mathbf{A} = \begin{Vmatrix} 29.50 & 26.00 & 28.75 & 26.50 & 15.50 \\ 26.00 & 24.25 & 28.25 & 26.00 & 17.50 \\ 28.75 & 28.25 & 35.75 & 30.50 & 25.00 \\ 26.50 & 26.00 & 30.50 & 30.00 & 20.00 \\ 15.50 & 17.50 & 25.00 & 20.00 & 22.00 \end{Vmatrix}.$$

Для нахождения собственных векторов матриц B и C, соответствующих максимальным собственным векторам и удовлетворяющих свойствам нормировки, использовался приближенный метод, который состоит в следующем:

$$B = \begin{Vmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{Vmatrix} \Rightarrow \vec{y} = \begin{Vmatrix} \sqrt[n]{b_{11} \quad b_{12} \quad \cdots \quad b_{1n}} \\ \sqrt[n]{b_{21} \quad b_{22} \quad \cdots \quad b_{2n}} \\ \cdots \quad \cdots \quad \cdots \quad \cdots \\ \sqrt[n]{b_{n1} \quad b_{n2} \quad \cdots \quad b_{nn}} \end{Vmatrix} \Rightarrow \vec{x} = \begin{Vmatrix} \frac{y_1}{\sum_{t=1,n} y_t} \\ \frac{y_2}{\sum_{t=1,n} y_t} \\ \cdots \\ \frac{y_3}{\sum_{t=1,n} y_t} \end{Vmatrix}. \qquad (7)$$

Произведя указанные вычисления, были получены значения весовых коэффициентов для каждого из показателей качества:

$$\vec{y} = \begin{Vmatrix} 31.70 \\ 21.73 \\ 25.56 \\ 39.21 \end{Vmatrix} \Rightarrow \vec{x} = \begin{Vmatrix} 0.268 \\ 0.184 \\ 0.216 \\ 0.332 \end{Vmatrix}.$$

Значения весовых коэффициентов распределились следующим образом:

F1 – устойчивость к механическим повреждениям – 0,268;

F2 – вибростойкость – 0,184;

F3 – простота сборки – 0,216;

F4 – патентная чистота – 0,332.

На четвертом этапе была проведена свертка имеющейся информации с целью выявления лучшей альтернативы. Множество оптимальных альтернатив *B* определялось путем пересечения нечетких множеств, содержащих оценки альтернатив по критериям

выбора. Так как критерии, по которым осуществлялся выбор альтернатив корпусов, имеют различную важность для лица, принимающего решение, то правило выбора лучшего варианта имеет вид:

$$B=F1^{\beta 1} \cap F2^{\beta 2} \cap ... \cap Fn^{\beta n};$$

$$\mu_B = (a^*) = \max \mu_B (a_j).$$

Здесь β_i определяется по формуле:

$$\beta_i = \alpha_i \, m \qquad (8)$$

где α_i – весовой коэффициент критерия F_i.

m – число критериев качества, по которым осуществляется выбор альтернатив корпусов.

Таким образом, оптимальной считается альтернатива с максимальным значением функции принадлежности к множеству B. Операция пересечения нечетких множеств соответствует минимальному значению j-й альтернативы:

$$\mu_B = (a_j) = \min \mu_{Fi} (a_j). \qquad (9)$$

Для рассматриваемой задачи множество оптимальных альтернатив приняло вид:

$B = \{\min\{0.2^{0.268}; 0.1^{0.184}; 0.2^{0.216}; 0.5^{0.332;}\}$
$\{\min\{0.8^{0.268}; 0.5^{0.184}; 0.5^{0.216}; 0.5^{0.332;}\}$
$\{\min\{0.8^{0.268}; 0.1^{0.184}; 0.7^{0.216}; 0.75^{0.332;}\}\}$.

Рассчитанный вектор приоритетов множеств оптимальных альтернатив принимает вид:

$\max \mu_B (a_j) = \max \{0.649; 0.794; 0.654\}$

Эта запись означает, что по результатам расчетов приоритеты между альтернативами распределились следующим образом:

Корпус 1 (настольный 19’’) – 0,649
Корпус 2 (монтаж в шкаф, 19’’) – 0,794
Корпус 4 (монтаж на стену) – 0,654

Следовательно, лучшей альтернативой является вторая – 19-дюймовый корпус для установки в шкаф (см. рис. 1), которой в векторе приоритетов соответствует значение 0,794.

Литература

1. А.А Добряков. Обеспечение творческих форм проектно-конструкторской деятельности в САПР силовых конструкций: Учеб. пособие для студентов высших технических учебных заведений. Москва ЦНИИНТИКПК 1989 г.
2. А.В Андрейчиков. Компьютерная поддержка изобретательства. М.: Машиностроение, 1998

Работа выполнена в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009 – 2013 годы

ВИРТУАЛИЗАЦИЯ В ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

Романова А.О.
Научный руководитель д. ф.-м. наук, профессор, Карпенко А.П.
МГТУ имени Н.Э.Баумана, Москва, Россия

VIRTUALIZATION IN HPC

Anastasia Romanova
Supervisor Karpenko A.P.
Moscow State Technical University N.A. N.E. Bauman

Аннотация
Рассмотрены преимущества виртуализации высокопроизводительных систем, дано краткое описание технологии виртуализации, приведены примеры использования технологии в современных HPC системах. Выполнено сравнение производительности реальной и виртуальной систем.

Abstract
This article discusses the benefits of virtualization of high-performance systems, a brief description of the virtualization technologies is given. There are examples of the use of technology in HPC systems. The text presents the results comparing the performance of the real and virtual systems.

Введение. Повышенный интерес к компьютерным технологиям виртуализации в настоящее время не случаен. Вычислительная мощь современных процессоров быстро растет. И вопрос даже не в том, на что эту мощь расходовать, а в том, что современные двухъядерные и многоядерные системы как нельзя лучше позволяют реализовать богатейший потенциал идей виртуализации операционных систем и приложений. Технологии виртуализации становятся одним из ключевых компонентов (в том числе, и маркетинговых) в новых процессорах Intel [1] и AMD [2], в операционных системах от Microsoft [3] и ряда других компаний. В данной статье рассматриваются идеи использования виртуализации в высокопроизводительных системах.

Типы виртуализации. Существует два больших класса типов виртуализации - аппаратная и программная виртуализации.

Под аппаратной виртуализацией понимают аппаратную поддержку, обеспеченную самим процессором. Аппаратная виртуализация – это общее название для двух независимых, но очень похожих технологий корпораций *Intel* и *AMD*, которые направлены на улучшение производительности процессора для общих задач виртуализации.

Программная виртуализация является виртуализацией в классическом понимании данного слова. Единой классификации видов данной програмной технологии не существует, однако наиболее полная таблица видов приведена в работе [4]. Ниже даны краткие характеристики некоторых типов програмной виртуализации, используемых в высокопроизводительных системах.

Серверная виртуализация (*полная виртуализация*), представляет собой полную эмуляцию оборудования на уровне программного обеспечения. Примером такой виртуализации является виртуальная машина (ВМ). ВМ — это окружение, которое представляется для «гостевой» операционной системы, как аппаратное. Однако на самом деле это программное окружение, которое эмулируется программным обеспечением «хостовой» системы. Именно аппарная виртуализация чувствительна к наличию аппаратной поддержки.

Данный подход имеет существенные накладные расходы, которые зависят от числа эмулируемых устройств и от степени их похожести на соответствующие физические устройства. Технология плохо масштабируется — нет возможности эффективно и совместно использовать все доступные аппаратные ресурсы. Оперативная память физической машины

фактически должна быть жестко разделена для того, чтобы предоставить каждой виртуальной машине свою собственную неразделяемую память. Кроме того, существенная доля оперативной памяти должна быть зарезервирована для накладных расходов самой системы виртуализации.

Другой особенностью данной технологии является жесткое разделение дискового пространства между ВМ — обычно, файловая система для виртуальной машины расположена внутри файла основной операционной системы (или раздела диска) и уникальна для каждого экземпляра ВМ.

Виртуализация уровня ОС (*виртуализация высокого уровня*; *совместная виртуализация* ОС) виртуализирует серверы на уровне ядра операционной системы [5]. При этом создаются изолированные «контейнеры» на одном физическом сервере и экземпляре ОС, чтобы обеспечить возможность максимально эффективно использовать программное и аппаратное обеспечение.

Виртуализация ОС рационализирована для достижения лучшей производительности, управляемости и эффективности. На нижнем уровне иерархии в этом случае лежит стандартная операционная система хост-компьютера (*Windows* или *Linux*). Далее идет слой виртуализации с собственной файловой системой, а затем - слой абстракции службы ядра, который обеспечивает изоляцию и безопасность ресурсов различных контейнеров. Контейнер является виртуальной средой исполнения — «высокоуровневой виртуализацией», которая обеспечивает каждой среде свое собственное уникальное изолированное окружение: файлы и системные ресурсы; сервисы; системные способы связи с хост-ЭВМ. Слой виртуализации позволяет каждому контейнеру функционировать, как самостоятельный сервер. Наконец, в самом контейнере размещается приложение или рабочий процесс.

По сравнению с серверной виртуализацией, высокоуровневая виртуализация дает большие преимущества: гибкое управление памятью и файловым пространством; низкие затраты на переключение между ВМ; возможность мгновенного получения всех ресурсов компьютера.

Паравиртуализация — техника виртуализации, при которой гостевые операционные системы подготавливаются для исполнения в виртуализированной среде, для чего их ядро незначительно модифицируется [6].

Цель изменения интерфейса заключается в сокращении доли времени выполнения операций в гостевой ЭВМ. Паравиртуализация, таким образом требует, чтобы гостевая операционная система была изменена для гипервизора. Это обстоятельство является недостатком метода, так как подобное изменение возможно лишь в случае, если гостевые операционные системы имеют открытые исходные коды, которые можно модифицировать согласно лицензии. В качестве компенсации, паравиртуализация обеспечивает производительность почти равную производительности реальной не виртуализированной системы. Как и при полной виртуализации, одновременно могут поддерживаться различные операционные системы.

Преимущества и проблемы виртуализации. Виртуализация вводит дополнительную гибкость в среду исполнения высокопроизводительных приложений. Такая гибкость может быть использована для решения следующих важных вопросов.

1. *Консолидация.* В рамках предприятия виртуализация в основном используется для сокращения числа физических серверов.
2. *Отказоустойчивость. В*иртуальные машины запускаются в четко определенном состоянии и не зависят от предыдущей работы.
3. *Безопасность* Виртуализация позволяет запускать несколько задач на одном физическом сервере и обеспечивает барьер для предотвращения утечки информации между процессами или пользователями.
4. *Портативность и управляемость.* Виртуализация позволяет выполнять различные смешанные задачи в отдельных виртуальных машинах на одном физическом сервере.

5. *Разработка и отладка.* Виртуализация может быть использована в "обратной консолидации" для отладки хорошо масштабируемых приложений по мере их разработки.

6. *Гетерогенность.* Технология виртуализации дает возможность реконфигурации узлов для нужд пользователей.

Несмотря на указанные достоинства виртуализации, остаются следующие вопросы, связанные с ней:

- могут ли виртуализированные системы обеспечить производительность, сравнимую с физическими системами?

- будет ли поведение программного обеспечения виртуализации и гостевой системы предсказуемым?

Виртуализация в высокопроизводительных системах. Использование виртуализации расширяет способы организации высокопроизводительных вычислений.

Облачные или *рассеянные вычисления* [7] представляют собой технологию обработки данных, в которой компьютерные ресурсы и мощности предоставляются пользователям как Интернет-сервисы. Для облачных вычислений основной проблемой является неравномерность запроса ресурсов со стороны клиентов. Для сглаживания этой неравномерности можно использовать промежуточный слой серверной виртуализации (часто виртуализации уровня операционной системы). Балансировка нагрузки осуществляется при этом, как средствами программного обеспечения, так и средствами распределения виртуальных серверов по физическим серверам.

Виртуализация в облаках позволяет получить без больших расходов доступ к потенциально очень большим вычислительным мощностям.

Грид-вычисления [8] представляют собой способ организации распределённых вычислений, в котором суперкомпьютер представлен в виде кластера слабосвязанных компьютеров, соединённых с помощью сети, и работающих вместе, для выполнения большого числа заданий. Эта технология применяется для решения научных задач, требующих значительных вычислительных ресурсов. Грид-технология может быть эффективно реализована на виртуальных кластерах.

Производительность. Главной причиной отказа от использования виртуализации, является потеря производительности виртуальной вычислительной среды по сравнению с физической средой. В то же время, имеются примеры, в которых эти потери несущественны. Так виртуальная машина Parallels позволяет пользователям выполнять вычисления на виртуальных графических процессорных устройствах (ГПУ) с производительностью, практически равной производительности физических ГПУ [9]. Такой результат достигается, благодаря использованию технологию *Intel VT-d*, которая позволяет напрямую назначать выделенные графические процессоры виртуальным машинам.

Исследования производительности *Grid*-системы, состоящей из виртуальных кластеров, выполнены в университете города Чикаго в 2009 году [10]. Исследования включали в себя эксперимент для оценки производительности OSG виртуального кластера при работе с различными классами приложений.

В ходе этих исследований сравнивалось время выполнения приложения на кластерах, состоящих из физических узлов, и на виртуальных кластерах, работающих на физических серверах. Для оценки производительности использовался тест *Fast Ocean Atmosphere Model* (*FOAM*) [11]. Результаты исследования иллюстрируют таблица 1 и рисунок 1.

Таблица 1 показывает, что разница в производительности виртуального и физического кластеров, которая не превышает 5%.

Таблица 1 -Потери производительности при использовании виртуального кластера

4 сервера	8 серверов	16 серверов
4,55%	4,05%	4,10%

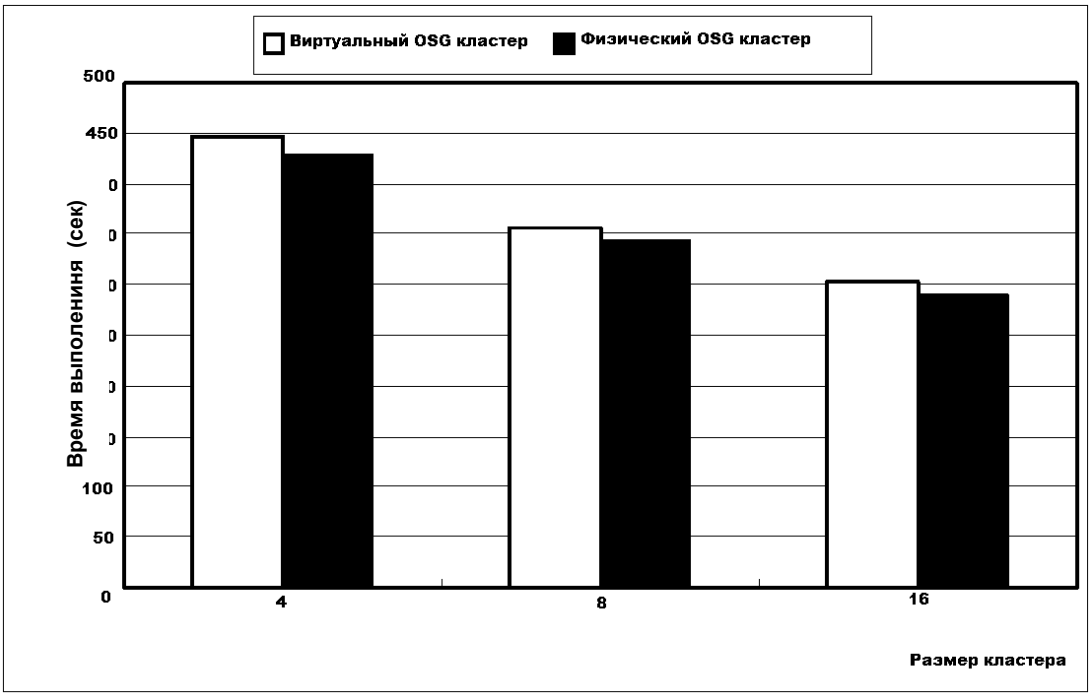


Рисунок 1 – К сравнению производительности виртуального и физического кластеров

Выводы. Виртуализация является гибким механизмом, помогающим в решении большого числа задач. Данная технология может быть широко использована, как для решения простых пользовательских проблем, так и в высокопроизводительных системах.

Литература

1. Технология Intel® Virtualization Technology for Directed I/O (Intel® VT-d). http://www.intel.com/cd/corporate/europe/emea/rus/update/360260.htm).
2. AMD Releases "Pacifica" Specification For AMD64. (http://www.amd.com/us/press-releases/Pages/Press_Release_98372.aspx).
3. Microsoft Hyper-V Server 2008 R2. (http://www.microsoft.com/hyper-v-server/en/us/default.aspx).
4. Виртуализация. (http://ru.wikipedia.org/wiki/Виртуализация).
5. Виртуализация ОС. (http://www.parallels.com/ru/products/pvc46/info/virtualization/).
6. Паравиртуализация.. (http://www.vmgu.ru/articles/Paravirtualizatsiya)
7. Об облачных вычислениях. (http://www.parallels.com/ru/spp/understandingclouds/)
8. Демичев А.П., Ильин В. А., Крюков А.П. Введение в грид-технологии: Препринт. - М.: НИИЯФ МГУ, 2007. -87 с.
9. Contain Your Cost, Control Your Grid. (http://www.parallels.com/products/extreme/audience/grid-computing/).
10. Virtual Clusters for Grid Communities. I. Foster, T. Freeman, K. Keahey, D. Scheftner, B. Sotomayor, X. Zhang. CCGrid, 2006.
11. R.Jacob, C. Schafer, I. Foster, M. Tobis, J. Anderson. Computational Design and Performance of the Fast Ocean Atmosphere Model, Version One. In International Conference on Computational Science. -Springer-Verlag.- 2001.

СОЗДАНИЕ ПРИЛОЖЕНИЯ ДЛЯ NX4 С ПОМОЩЬЮ СРЕДСТВ NX OPEN API

Ныннырова А.С.
Научный руководитель: Доцент кафедры РК6 , к.т.н. Мартынюк В.А.
МГТУ им. Баумана кафедра РК6, Москва, Россия

Nynnyrova A.S.
Associate Professor, PhD, lecturer RK6 Martyniuk V.A.
MSTU n.a. Bauman, Moscow, Russia

Аннотация
В тексте доклада излагаются правила и трудности написания приложений для системы NX с помощью библиотек NX Open. Особое внимание уделяется той части приложений, в которых приходится иметь дело со сложными сборками. По этим правилам было написано приложение для фирмы «Иркут» для расстановки заклепок на обшивке проектируемого среднемагистрального самолета МС21.
Abstract
The aim of this paper is to explore the function libraries NX Open, to write applications for NX4. As part of the work has been studied in detail some basic library functions NX Open for working with parts and assemblies. And has been written an application that allows you to automate the placement of rivets. The size of rivets is adjusted for the thickness of the package and the diameter of the hole.

NX - это интерактивная система автоматизации проектирования и изготовления. Для обозначения систем этого класса используется аббревиатура CAD/CAM (Computer-Aided Design и Computer-Aided Manufacturing). Подсистема CAD предназначена для автоматизации проектных, конструкторских и чертежных работ на современных промышленных предприятиях. NX Open – модуль NX, с помощью которого можно программным способом выполнять практически все функции, доступные пользователю при интерактивном взаимодействии с системой. То есть, NX Open – это и есть тот инструмент, с помощью которого пишутся приложения для системы NX.

Для компилирования приложений NX в операционной системе WIN IA32 (WNTI32) сертифицирована программа *Microsoft Visual Studio 2003*. Для создания приложения удобно воспользоваться шаблоном проекта, который идет вместе с NX, его можно найти в папке *Program Files\UGS\NX 4.0\UGOPEN*. Для этого две папки *vcprojects* и *VCWizards* необходимо скопировать в директорию Program *Files\Microsoft Visual Studio .NET 2003\Vc7*.

Инструменты NX Open позволяют выбрать язык приложения. В нашем случае предпочтение было отдано языку C.

Использование Open User Interface Styler

С помощью приложения *Open User Interface Styler* можно автоматически создавать оконные приложения и получать шаблоны для дальнейшего написание программы. Панель инструментов состоит из объектов, которые можно вставит в создаваемое диалоговое окно и стандартных команд управления файлами – создать, сохранить, открыть, вставить, вырезать.

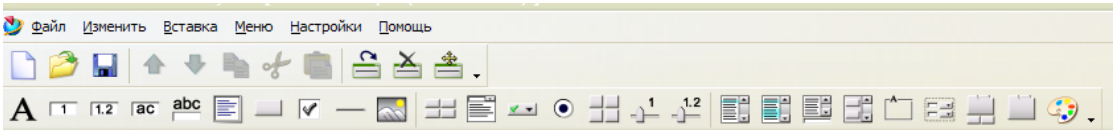


Рисунок 1 - Панель инструментов в режиме создания интерфейсов

Для создания внешнего вида диалогового окна нужно просто выбрать объекты и поместить их в требуемом порядке в окно проектируемого диалога.

Диаграмма внизу показывает отношения между объектами *Open User Interface Styler*. Создаваемый диалог (*Dialog*) состоит из нескольких пунктов-объектов *(Dialog Item*) (кнопки,

меню выбора, окошки для ввода чисел или строк и т.д.), каждый из которых имеет свои ресурсы (*Resource*) – параметры и настройки, которые в свою очередь состоят из атрибутов (*Attribute*) и функций вызова (*Callbacks*).

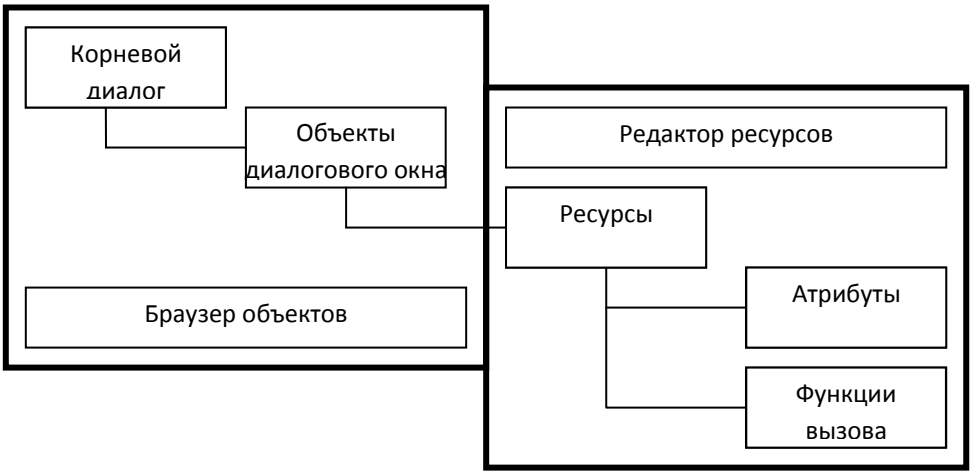


Рисунок 2 - Отношения между объектами *Open User Interface Styler*

Корневой диалог также как и все его объекты имеет свои атрибуты. Каждый объект имеет свой набор атрибутов, но у всех объектов есть такие атрибуты как *Метка* и *Идентификатор*. Для объектов и корневого окна задаются функции вызова, они будут выполняться при совершении определенных действий с объектом.

После окончания создания объектов и задания их параметров необходимо сохранить диалоговое окно. При этом система может попросить указать язык для шаблонов, в нашем случае C. При сохранении генерируются три файла с расширениями: *dlg* – бинарный файл, содержащий внешний вид диалогового окна, *_template.c* – C файл, содержащий функции пользовательского входа и шаблоны функций вызова, *h* – заголовочный файл, содержащий определения функций вызова.

Файл с расширением *dlg* необходимо поместить в папку *Application*, а путь к этой папке прописать в файле *custom_dirs.dat*, который находится в папке *C:\Program Files\UGS\NX 4.0\UGII\menus*. Например если папка *Application* находится в директории *C:\UG*, то в конец файла с новой строки нужно дописать *C:\UG*. В эту же папку *Application* необходимо поместить полученный в самом конце *dll* файл динамически загружаемой библиотеки.

Основной тип данных NX Open

Важно отметить – как в системе и в приложении идентифицируются графические примитивы и детали сборки. Каждый объект NX имеет свой идентификатор – тэг (*tag*). В заголовочном файле *uf_defs.h* определяется тип данных *tag_t*, основанный на *unsigned int*, он присваивается как порядковый номер каждому объекту, созданному *Unigraphics* в данной модели. Существует специальное значение тэга – *NULL_TAG*, которое никогда не присваивается объектам. *NULL_TAG* используется для отслеживания ошибок создания объекта или для поиска. С точки зрения его применения *NULL_TAG* напоминает указатель *NULL*.

Итак, каждый загруженный в течение сессии объект имеет уникальный тэг. Тем не менее тэг присвоенный объекту не постоянен ни в течение нескольких сессий NX, ни при множественном использовании одной детали в одной сессии. Тэг может быть использован повторно в течение одного запуска NX. Например, если часть геометрии создана и впоследствии удалена, то ее тэг далее может быть использован для идентификации другой геометрии.

Основные функции NX Open

Функция *ufusr* используется как точка входа, она представляет собой функцию main для программы создания внутреннего приложения NX. Как правило, эта main функция только высвечивает диалоговое окно вашего приложения. Функция ufusr имеет три формальных параметра. Первый параметр – входной, используется, если эта функция вызывается из *Menuscript* (специальное приложение для создания собственных и изменения уже существующих меню в NX) и содержит код выбранной кнопки меню. Второй параметр является выходным и содержит код возврата, этот параметр не используется во внутренних приложениях. Третий параметр должен содержать длину первого параметра.

Функция *UF_initialize* инициализирует среду Open C API и загружает лицензию выполнения Open API. Эта функция должна вызываться сразу после объявления всех локальных переменных, если этого не сделать, то все вызываемые далее функции будут возвращать ошибку *UF_err_program_not_initialized*. Функция *UF_terminate* «уничтожает» среду Open C API и выгружает лицензию выполнения Open API, если вызвана эта функция, то программа должна завершиться, нельзя опять вызвать *UF_initialize* и продолжить работу. Обе эти функции возвращают код ошибки и вызываются без параметров.

В функции *ufusr_ask_unload* с помощью возвращаемого значения можно определить способ выгрузки DLL библиотеки после выхода из приложения. Для этого предусмотрено три предопределенных константы: 1) UF_UNLOAD_IMMEDIATELY – выгрузка сразу после выполнения приложения, 2) UF_UNLOAD_SEL_DIALOG – явная выгрузка библиотеки при помощи диалога *Выгрузка динамически загружаемой библиотеки*, 3) UF_UNLOAD_UG_TERMINATE – выгрузка при завершении работы NX.

Вместе с функцией *ufusr_ask_unload* используется функция очистки перед завершением приложения - *ufusr_cleanup.* Она вызывается непосредственно перед выгрузкой DLL библиотеки. Например, если при выполнении приложения вызывались какие-то файлы базы данных, то функцию *ufusr_cleanup* можно использовать для их закрытия.

Описание работы программы

Интерфейс нашего конкретного приложения представляет собой диалоговое окно с двумя кнопками «Выберите грани пакета» и «Выберите отверстие», ниже находится меню выбора «выберите заклепку», далее три стандартные кнопки «ОК», «Применить» и «Отмена».

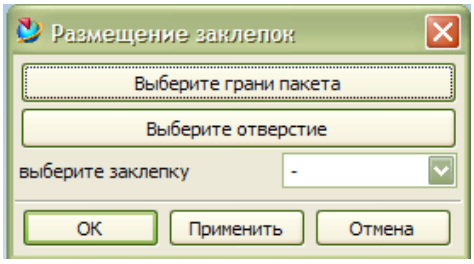


Рисунок 3 - Интерфейс созданного приложения.

Последовательность выбора крайних граней, отверстия и заклепки любая. Нажав на кнопку «Выберите грани пакета» необходимо выбрать две крайних грани пакета, при этом запоминаются указатели на эти плоскости и находится расстояние между ними.

Нажав на кнопку «Выберите отверстие» нужно выбрать отверстие под заклепку, при этом запоминается указатель на него и находится его радиус кривизны.

Из меню выбора необходимо выбрать тип заклепки при этом устанавливается имя файла с прототипом выбранной заклепки.

Когда все значения установлены, нажмите на кнопку «Применить» при этом открывается файл детали с выбранной заклепкой, изменяется ее длина и диаметр под толщину пакета и диаметр отверстия, новая деталь сохраняется под новым именем, вставляется в сборку и сопрягается по плоскости и отверстию. Для следующей заклепки не обязательно задавать все условия установки, если какой-либо параметр не будет задан, то считается, что он остается неизменным с предыдущего раза. При нажатии кнопок «Отмена» или «ОК» приложение закрывается.

Литература

1. Библиотека справочников NX 4.
2. М. Краснов, Ю. Чигишев. Unigraphics для профессионалов. Лори, 2004 г. 320 стр.
3. П. Гончаров, М. Ельцов, С. Коршиков, И. Лаптев, В. Осиюк NX для конструктора-машиностроителя. ДМК Пресс, 2010 г. 504 стр.

РАЗРАБОТКА ПРОГРАММНО-АППАРТНОГО КОМПЛЕКСА ДЛЯ РЕГИСТРАЦИИ ПЕРЕГРУЗОК БОЛЬШЕГРУЗНЫХ АВТОМОБИЛЕЙ

Григорьев В.А.
Научный руководитель доцент Попов А.Ю.
МГТУ им.Н.Э. Баумана, г.Москва, Россия

DEVELOPMENT OF HARDWARE AND SOFTWARE COMPLEX FOR REGISTRATION OF OVERLOADS OF TRUCKS

Grigoryev V.A.
Prof. Popov A.Yu.
BMSTU, Moscow, Russia

Аннотация
Работа посвящена разработке программно-аппартного комплекса для регистрации перегрузок большегрузных автомобилей. Представлена схема комплекса, основные принципы работы.
Abstract
This work is dedicated to developing of hardware and software complex for registration of overloads of trucks. A scheme of the complex, the basic principles of operation.

В настоящее время существует проблема быстрого износа большегрузных автомобилей. Это связано с тем, что при эксплуатации грузовых автомобилей не всегда соблюдаются правила эксплуатации, установленные в технической документации на эти автомобили. Для получения дополнительной прибыли превышается максимально допустимый вес перевозимых грузов, что является основным фактором, приводящим к износу техники.

В связи с этим возникает задача контроля массы перевозимых грузов с использованием специальной измерительной аппаратуры, которая будет сигнализировать в случае превышения нормы загрузки автомобиля и записывать информацию о весе грузов в энергонезависимую память.

Для решения поставленной задачи необходимо разработать комплекс, состоящий из совокупности датчиков угловых положений и линейных ускорений, ЖК-дисплея и вычислительного устройства, обрабатывающего показания с датчиков и записывающего итоговую информацию в энергонезависимую память.

Цель работы: разработка схемы комплекса, подбор составляющих комплекса, разработка программного обеспечения.

При разработке схемы необходимо руководствоваться следующим: основной блок вычислительного устройства должен располагаться в кабине водителя на приборной панели. Ко входу питания будет подводится 24 В от аккумулятора автомобиля. На приборной панели устройства должен присутствовать CAN-разъём для подсоединения датчиков углового положения и линейного ускорения и COM-разъем для считывания информации из Flash-памяти устройства.

Разработанная функциональная схема представлена на рисунке 1. Комплекс состоит из источника бесперебойного питания 2, вторичного источника питания 3, лицевой панели комплекса 4, микроконтроллера 5, оконечного оборудования CAN 6, Flash памяти 7, GPS/GSM модуля 8, и контрольно-измерительных датчиков 9.1 – 9.n., резервного аккумулятора 10, сети CAN 11, шин питания 24В 12,12В 13, 5В 14, 3.3В 15, информационных шин контрольно-измерительных датчиков 16.1-16.n i, информационной шины источника питания 17 pow, информационной шины зуммера 18 cnt, информационной шины индикаторов 19 cnt, информационной шины кнопки 20 but, информационной шины

дисплея 21 disp, информационной шины COM 22 us, шины Flash-памяти 23 mem, шины GPS/GSM модуля 24 d,t, блока управления и индикации 25.

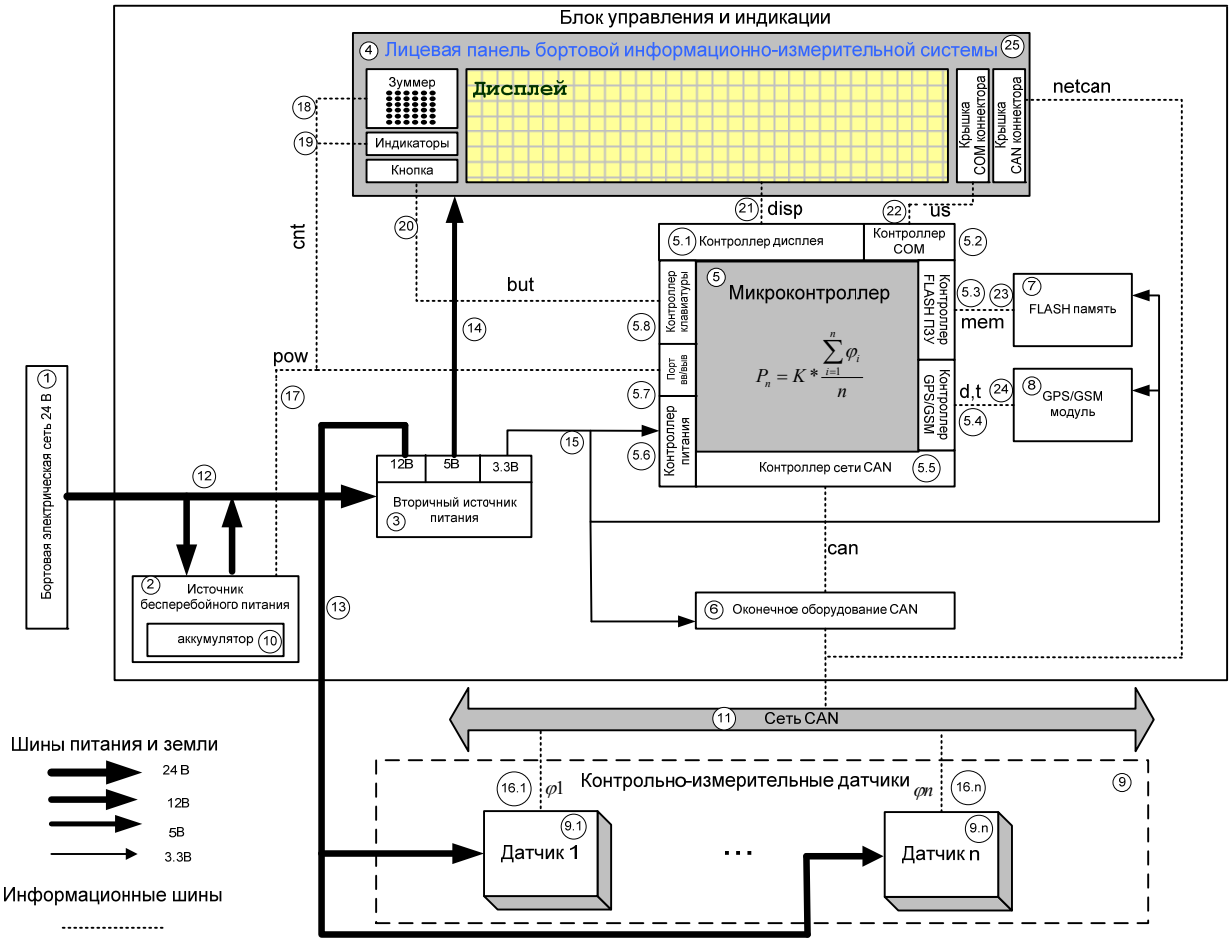


Рисунок 1 - Функциональная схема комплекса

Напряжение бортовой электрической сети 1 по шине питания 24В поступает на ИБП 2 с резервным аккумулятором 10. Напряжение питания 3,3В с ВИП 3 поступает через шину питания 15 на контролер питания 5.6 МК 5, и на цепи питания Flash-памяти 7, GPS/GSM модуля 8 и оконечного оборудования CAN 6. При поступлении по шине питания 24В необходимого напряжения аккумулятор 10 заряжается, а при понижении необходимого уровня аккумулятор 10 поддерживает уровень напряжения питания в 24В для корректного завершения работы комплекса.

Информация с контрольно-измерительных датчиков 9.1 – 9.n в кодированном виде цифровых значений углового положения φi по информационным шинам 16.1 – 16.n поступает в сеть CAN 11 и из нее в контроллер сети CAN 5.5 и CAN разъем лицевой панели 4.6. Информация о текущем географическом положении и о времени поступает с GPS/GSM модуля 8 по шине 24 d,t в контроллер GPS/GSM 5.4 МК 5. Информация от ВИП 3 поступает на порт ввода-вывода 5.7 МК 5 по шине 17 pow. Сохранение информации происходит по шине 23 во Flash-память 7.

Информация с контрольно-измерительных датчиков 9.1-9.n, с GPS/GSM модуля 8 и кнопки 4.3 обрабатывается в МК 5, анализируется и данные сохраняются в энергонезависимой Flash-памяти 7 по шине 23.

Разъем подключения комплекса к бортовой электрической сети PC19, Intro CH. Вторичный источник питания 3**,** на основе аккумуляторной батареи 10, обеспечивающий полноценное завершение работы комплекса. Данный источник обеспечивает работу системы при потере питания 24 В. Лицевая панель комплекса 4**,** с установленным на ней дисплеем

4.4, зуммером (динамиком) 4.1, индикаторами (4.2), кнопкой управления 4.3, разъемами COM 4.5 и CAN 4.6, прикрытых крышками, предназначена для вывода текущей информации об автомобиле.

Оконечное оборудование CAN 6 – микросхемы, обеспечивающие организацию сети CAN. GPS/GSM модуль – модуль, обеспечивающий географическое позиционирование и точное текущее время. Контрольно-измерительные датчики – набор из N датчиков углового положения с интерфейсом CAN.

Работа комплекса заключается в следующем: при включении зажигания в автомобиле происходит подача напряжения питания от бортовой системы автомобиля через разъем 1 на шину питания 24В 12. Напряжение 24В подается посредством шины питания 12 на ИБП 2 и ВИП 3. ИБП 2 имеет в своем составе аккумулятор 10, способный обеспечить корректное завершение работы комплекса через шину 12.1. Аккумулятор 10 заряжается в процессе работы комплекса через шину 12.2. ВИП 3 устанавливает на шине 13 напряжение 12В, на шине 14 напряжение 5В, на шине 15 напряжение 3.3 В. Далее на МК 5 включается процесс самотестирования, заключающегося в проверке работоспособности контрольно-измерительных датчиков 9.1-9.n и Flash-памяти. При успешном прохождении процесса самотестирования происходит индикация этого успеха на дисплее 4.4 лицевой панели 4.

Затем на контроллер сети CAN 5.5 МК 5 поступает информация с контрольно-измерительных датчиков 9.1-9.n в цифровом виде об угловом положении каждого контрольно – измерительного датчика по сети CAN 11. Каждый контрольно-измерительный датчик расположен на рессоре согласно рис.2. и сориентирован на рессоре так, что (рис.3): ось OX – лежит в плоскости местного горизонта и совпадает с продольной осью симметрии рессоры и направлена в положительном направлении движения автомобиля; ось OY – лежит в плоскости местного горизонта и перпендикулярна оси X; ось OZ – перпендикулярна плоскости местного горизонта и направлена к центру Земли.

В МК 5 производится обработка полученных сигналов от контрольно-измерительных датчиков 9.x для последующего вычисления загруженности рессор автомобиля по формуле:

$$P_m = K * \frac{\sum_{i=1}^{n} \varphi_i}{m},$$

где φ_i – цифровое значение угла наклона i-го контрольно-измерительного датчика, выдаваемое им по сети CAN 11 за вычетом значений датчика 9.n, K – коэффициент зависимости угла отклонения рессоры от нагрузки на нее, m – количество контрольно-измерительных датчиков на рессорах.

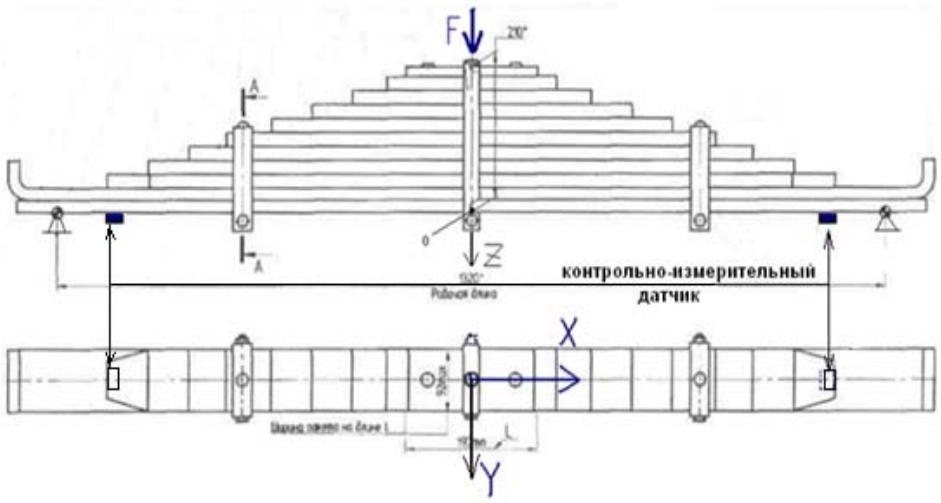


Рисунок 2 - Схема расположения контрольно-измерительных датчиков на рессоре

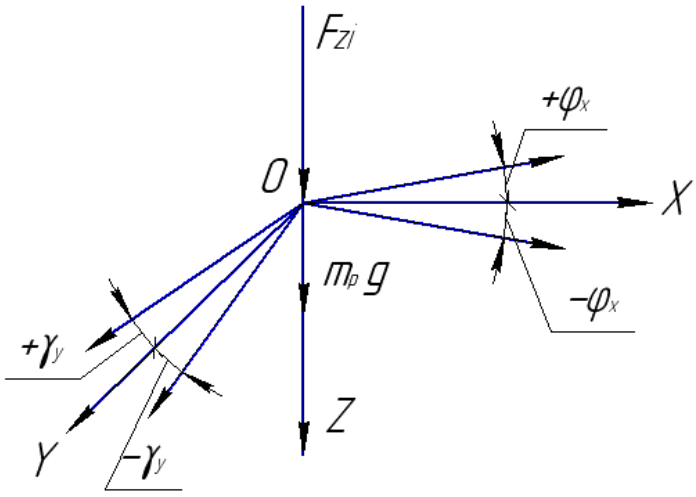


Рисунок 3 - Ориентация контрольно – измерительного датчика на рессоре

При превышении массы общего количества груза, перевозимого автомобилем, комплекс помечает это событие в файле во Flash-памяти 7 в месте, соответствующем перегруженной рессоре и в колонке, соответствующей массе общего количества груза. Также раздается звуковой сигнал в кабине водителя посредством зуммера 4.1. Информация о загруженности осей автомобиля будет сохраняться в энергонезависимой Flash-памяти 7. В ней будут находиться файлы, создающиеся автоматически с началом нового дня при первом включении зажигания.

Литература

1. Патент 2162202, РФ, G01C7/04, E01C23/07.Способ определения уклонов, кривизны, неровности и коэффициента сцепления дорожного покрытия и устройства его осуществления/ Ачильдиев В.М.; Дрофа В.Н.; Рублев В.М.; Сорокин В.Е.; Цуцаев Д.А.; опубл. 20.01.2001

МЕТОД МИНИМИЗАЦИИ ПРОСТРАНСТВА ИНФОРМАЦИОННЫХ ПРИЗНАКОВ В ЗАДАЧАХ ТАКСОНОМИИ

Краснов А.С.
Научный руководитель: к.т.н., доцент кафедры РК6, Волосатова Тамара Михайловна
МГТУ им. Н.Э.Баумана, Москва, РФ

Krasnov Anton Sergeevich
Scientific adviser: c.t.s., associate professor, Volosatova Tamara Mihajlovna
MSTU named after N.E.Bauman, Moscow, Russian Federation

Аннотация
В представленном докладе рассмотрена тема минимизации пространства информационных признаков большой размерности. Основная задача исследования – разработать метод, позволяющий быстро минимизировать признаковое пространство, сохраняя, вместе с тем, репрезентативность и полноту этого пространства.
Abstract
This paper highlights the issue of high dimensionality information attributes space minimization. The main task is to develop a method that quickly minimizes the attributes space, preserving, however, the representativeness and completeness of this space.

Очевидно, что одна и та же система признаков может быть информативной для решения одной задачи распознавания и не информативной для другой. Оценка информативности признаков зависит от того, что от чего нужно отличать, т.е. от списка распознаваемых образов $S = <s_1, s_2,...s_i,...s_k>$. Зависит она и от типа решающих функций D. Так что, указать "типичные" или "часто используемые" признаки невозможно. Для каждой задачи, для каждой обучающей выборки A нужно находить свое информативное множество описывающих ее признаков X_n.

«Информативность признаков» - понятие относительное. Этот термин, часто используемый в различных публикациях до настоящего времени по-разному трактуется разными авторами. В данной работе предлагается определить «информативность признака» I^f_i, как частоту повторяемости $s(i)$ признака i к общему числу признаков N исследуемого признакового пространства:

$$I_i^f = \frac{s(i)}{N}$$

Пусть дан некоторый класс распознаваемых образов $S = <s_1, s_2,...s_i,...s_k>$. Для каждого из классов выделим набор признаков $X_i = <x_{i1}, x_{i2}, ..., x_{in}>$ и зададим порог информативности θ. Также, у нас имеется обучающая выборка A, включающая в себя некоторое количество элементов каждого класса распознаваемых объектов.

Для начала, попробуем классифицировать признаки по степени их важности от самого важного к самому незначительному. Будем использовать нейронную сеть, построенную на базе нейронов типа Гроссберга. Прежде, соберем все предложенные наборы признаков в единое признаковое пространство размерности p, не забывая исключать дублирующиеся признаки: $\Omega^p = <x_{11}, x_{12}, ..., x_{i1}, x_{i2}, ..., x_{j1}, x_{j2}, ..., x_{k1}, x_{k2}..., x_{kp}>$. На первом слое сети расположим p нейронов, по числу предлагаемых классифицирующих признаков x пространства Ω^p. Каждому нейрону зададим такую пороговую функцию φ_i, что если признак x_{ji} в предъявленном на вход сети образце a_{in} будет выявлен, то функция увеличит вес признака. Если же признак не проявился – то уменьшит его вес соответственно. Проверив все экземпляры класса s_1, соберем статистику весов, получив набор $X^0_1 = <x_{11}, ..., x_{32}, ..., x_{mn}>$, состоящий из тех признаков, частота повторяемости которых не выше заданного порога θ. Стоит отметить, что в данном случае возможны ситуации, когда в набор лучших признаков класса могут попасть признаки, имевшие отношение к совершенно иному классу объектов. Таким образом, мы позволяем проводить анализ признакового пространства не с

точки зрения априорной единственности распознаваемого класса, но и с точки зрения групповой принадлежности признака сразу к нескольким объектам различных классов. После анализа сбросим весовые коэффициенты, и начнем проверку следующего класса s_2. Такую проверку будем проводить для всего набора распознаваемых классов множества S.

После того, как все экземпляры всех классов изучены, будем иметь набор наилучших признаков X^0_i для каждого класса s_i. Теперь, составим таблицу, где строки будут являться признаками, отобранными на первом этапе. В качестве столбцов будем использовать классы исследуемых объектов. Для простоты изложения положим, что число исследуемых классов равно 5, а количество отобранных признаков равно 8.

Таблица 1 - Матрица, отображающая связи между классами анализируемых объектов и совокупным набором признаков этих классов (0 опущены)

	s_1	s_2	s_3	s_4	s_5
x_1	*1*	*1*			*1*
x_2		*1*	*1*		*1*
x_3	*1*			*1*	
x_4				*1*	*1*
x_5		*1*			
x_6	*1*		*1*	*1*	
x_7		*1*	*1*		*1*
x_8	*1*				

На следующем этапе алгоритма наша задача – минимизация пространства признаков. Для того чтобы решить данную задачу и найти минимальный набор признаков, воспользуемся строчным покрытием полученной булевой матрицы методом Закревского. Алгоритм этого метода заключается в следующем:

1. Ищется столбец с минимальным числом единиц. Если таковых несколько, то выбирается любой (для определенности, допустим, самый левый).

2. Среди строк, покрывающих этот столбец, ищется строка с максимальным числом единиц и заносится в покрытие (следовательно, удаляется из матрицы); если же таких строк несколько, то выбирается любая из них (для определенности, допустим, самая верхняя).

3. Удаляются все столбцы, которые покрывает полученная строка.

Алгоритм продолжает свою работу до тех пор, пока не будет удалена вся матрица связей признаков.

Если выполнить предложенный алгоритм для таблицы 1, то:

Итерация 1: строка x_2 добавлена в минимальное покрытие, столбцы s_2, s_3, s_5 удалены;

Итерация 2: строка x_3 добавлена в минимальное покрытие, столбцы s_1, s_4 удалены.

Таким образом, одно из возможных минимальных покрытий указанной матрицы – $\{x_2, x_3\}$. Для большей объективности можно повторить алгоритм, но теперь выбирать из всех столбцов с одинаковым числом единиц не самый левый, а самый правый, получив, таким образом, другой набор покрытия – $\{x_5, x_6, x_7\}$.

Благодаря высокой скорости работы этого алгоритма даже на матрицах большой размерности (при соответствующих модификациях исходного кода алгоритма), можно выполнять быструю минимизацию признакового пространства, не волнуясь о вычислительных расходах. На представленном ниже графике, показаны результаты работы предложенного алгоритма по сравнению с другими существующими алгоритмами.

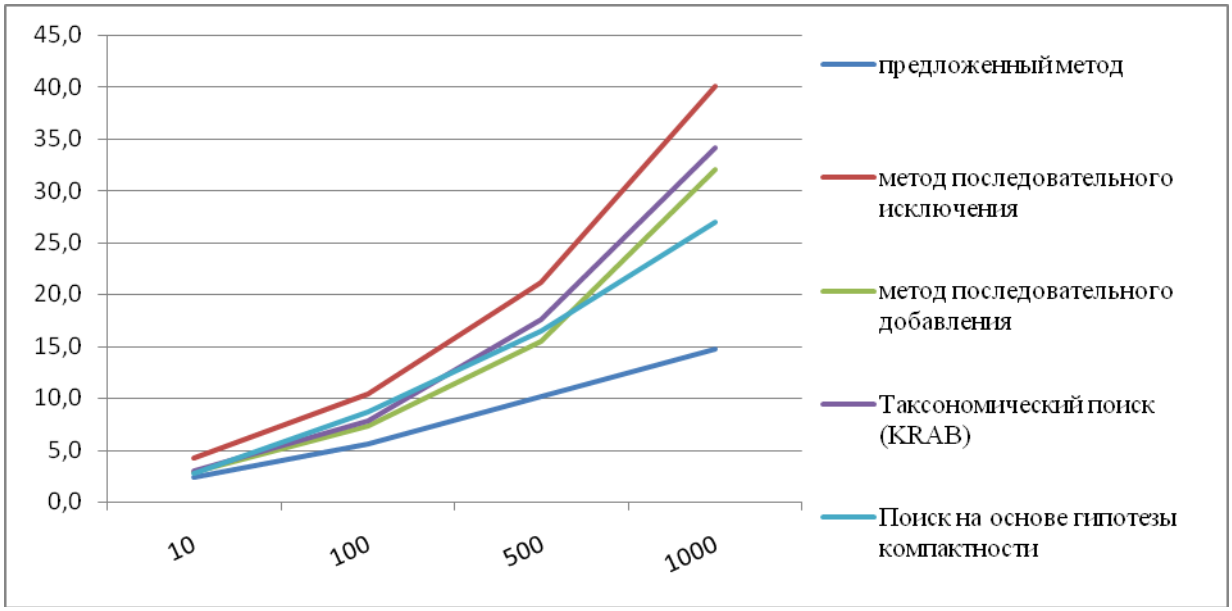


Рисунок 1 - Сравнение времени работы различных методов оптимизации в зависимости от размерности пространства информационных признаков

Из приведенной таблицы видно, что предложенный алгоритм имеет хорошее быстродействие при работе с пространствами признаков большой размерности.

Литература

1. Распознавание образов в массивах взаимосвязанных данных, С.Д. Двоенко, ММРО-10, с. 37-40
2. Optimization by simulated annealing, Kickpatrick S., Gelatt C., M. Vecci, 1983, pp 671-680
3. О некоторых подходах к вычислению информативных характеристик обучающей выборки, Е.В. Дюкова, Н.В. Песков, ММРО-9, 1999, с.181-183

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ФОРМИРОВАНИЯ И ОБРАБОТКИ СТЕРЕОИЗОБРАЖЕНИЙ

Горин Я.А., Марченков А.М.
Научный руководитель к.т.н. Волосатова Т.М.
Кафедра РК6, МГТУ им. Н.Э. Баумана, Москва, Россия

SOFTWARE DEVELOPMENT FOR PRODUCING AND PROCESSING OF STEREO

Gorin J.A., Marchenkov A.M.
The scientific chief: Ph.D. senior lecturer Volosatova T.M.
Department RK6, BMSTU, Moscow, Russia

Аннотация
В работе приведен обзор задач формирования и обработки стереоизображений. Рассмотрены технические особенности и возможные улучшения реализованного алгоритма. Разработано ПО для формирования и обработки стереоизображений. Проиллюстрированы результаты работы ПО.
Abstract
The article provides an overview of the tasks of generation and processing of stereo images. The technical features and the possible development of the algorithm are considered. Software for generation and processing stereo images was developed. The article is illustrated by the program output.

Видео и фотоизображения тесно вплелись в нашу жизнь. Почти каждый мобильный телефон оснащен камерой. Почти каждая камера умеет записывать видеоизображения. Повсеместно распространилась 3D-графика. С развитием возможностей усиливается потребность в “дешевом” построении 3D-сцен. Самый очевидный из таких методов — стереозрение — получение трехмерной картины мира по видеоряду или нескольким изображениям.

Совмещение изображений, позволяет человеку получить информацию о расстоянии до объектов по их расхождениям. Эта идея может быть использована в алгоритмах обработки изображений. Хотя, конечно же, зрение человека активное (то есть параметры оптической системы настраиваются под изображение) — глаза вращаются в глазницах, меняется фокусное расстояние, как правило, построение системы активного зрения сложнее и дороже, чем зрения пассивного. Система пассивного стереозрения, как правило, включает в себя 2 камеры. Существуют различные классификации алгоритмов сопоставления 2-х. Один из вариантов такой классификации представлен в [2].

Исходные предположения. Большинство алгоритмов подразумевают некоторые исходные предположения об окружающем мире. Наиболее распространенными являются**:**

•Ламбертовость поверхностей.

•Поверхности в реальном мире кусочно-гладкие.

•Предполагается, что камеры откалиброваны.

•Ограничение упорядоченности (order constraint). Это значит, что если на левом изображении точки идут слева направо в каком-то порядке, то в правом изображении они следуют в том же порядке (рис. 1).

•Для корректной работы большинства алгоритмов требуется ректификация стереопары, то есть выполнение преобразования, при котором и правое, и левое изображение проецируются на плоскость, параллельную базовой линии (линии, соединяющей оптические центры j, объективов камер) (рис. 1).

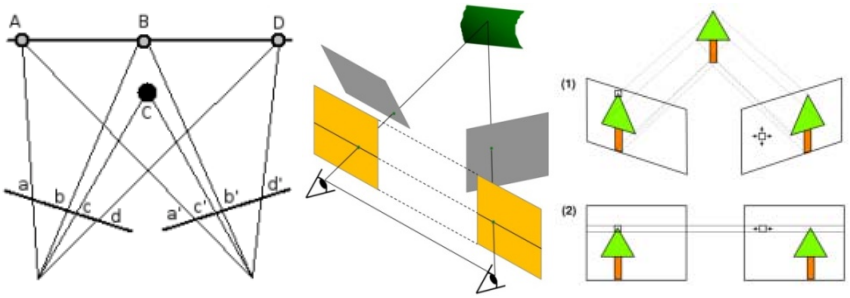


Рисунок 1 - Нарушение ограничения упорядоченности (слева) и ректификация (справа).

Соответствие границ интенсивности и резких перепадов функции расхождения. Это значит, что на границах объектов интенсивность меняется достаточно резко.

Почти все алгоритмы предполагают, что каждому пикселю одного изображения соответствует пиксель другого, в противном случае он считается заслоненным.

Существует множество подходов и алгоритмов стерео сопоставления. Подробный обзор доступен на странице Middlebury Stereo Vision [10].

Сведение видеоизображений с разных веб-камер. Теоретически возможно использование разных камер, но при этом может возникнуть ряд трудностей. Эксперимент, проведенный с тремя разными веб-камерами, демонстрирует возникающие проблемы.

В эксперименте было использовано следующее оборудование: встроенная веб-камера ноутбука Dell Studio 1535, Logitech QuickCam IM/Connect и Logitech QuickCam Pro 4000. Схема эксперимента показана на рисунке 2. Съемка производилась при одинаковой освещенности. Для всех камер было установлено разрешение 640x480 и цветовое пространство RGB.

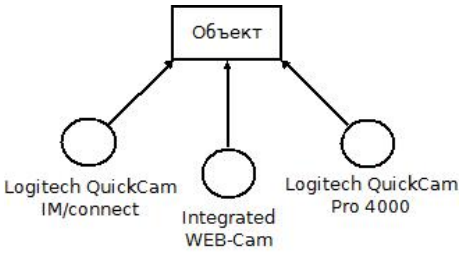


Рисунок 2 - Схема эксперимента.

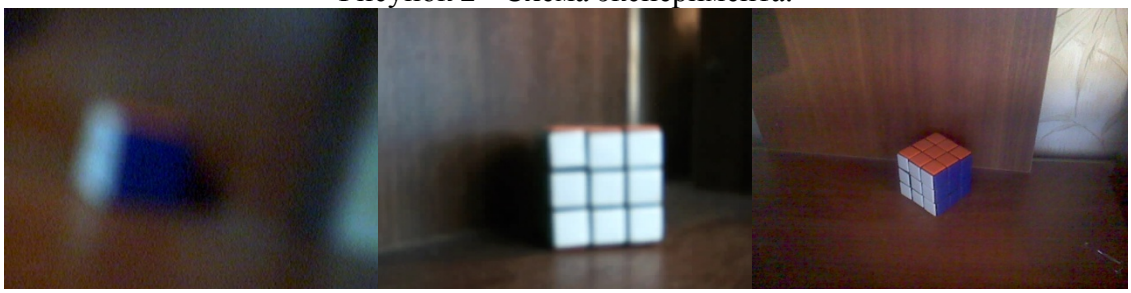


Рисунок 3 - Снимки камер Logitech QuickCam Pro 4000, Logitech QuickCam IM/Connect, со встроенной веб-камеры ноутбука Dell Studio 1535.

Самое качественное изображение получено со встроенной веб-камеры рис. 2, оно резкое и контрастное, объект практически не зашумлен. Изображение с камеры Logitech QuickCam IM/Connect не зашумлено, но дефокусированно. Самый некачественный снимок получен с камеры Logitech QuickCam Pro 4000, он сильно зашумлен и дефокусирован.

Из результатов эксперимента видно, что без предварительной обработки полученные изображения использовать нельзя. Так как, чтобы использовать снимки для нахождения общих деталей, они должны быть близкими по качеству. Иначе невозможно будет выбрать критерий, по которому будет производиться сравнение. В качестве предварительной

обработки как минимум нужно будет использовать операции по устранению дефокусировки, смаза и зашумленности. Все эти операции требуют больших затрат ресурсов как машинных, так и временных. Даже при одинаковых настройках качество снимков очень сильно отличается. Что приводит к значительным дополнительным затратам и вызывает значительные затруднения при сведении стереоизображений. Поэтому использовать разные камеры нецелесообразно.

Калибровка камеры – необходимый шаг в 3D компьютерном зрении, чтобы извлечь метрическую информацию из 2D изображений. Калибровку можно классифицировать на две категории: фотограмметрическая калибровка и само-калибровка.

Фотограмметрическая калибровка. В этом случае калибровка камеры делается с помощью наблюдения калибрующего объекта, чья геометрия в 3D пространстве известна с очень хорошей точностью. Также используется точно знаемый сдвиг переносимой плоскости. Подходы требуют дорогую калибрующую аппаратуру и тщательную установку.

Само-калибровка. При этом способе калибрующий объект не используется. При движении камеры в статичной сцене, жесткость, которой обеспечивают два ограничения на внутренние параметры камеры из одного сдвига, используют только информацию единственного изображения. Этот подход достаточно гибок, но не доработан. В нем много параметров для оценивания, поэтому в ряде случаев результаты неверны.

Можно объединить предыдущие два способа калибровки. Подобранный метод требует только камеры для съемки плоского объекта, показанной под несколькими (минимум двумя) разными ориентациями. Либо камера, либо объект может двигаться. Движение не известно. Этот способ лежит между фотограмметрической калибровкой и само-калибровкой. По сравнению с классической и само-калибровкой, эта техника значительно более гибкая и добавляет значительную степень надежности.

Таким образом процедура калибровки следующая:

1. Напечатать модель и прикрепить к плоской поверхности.
2. Снять несколько изображений модельной плоскости, под разными направлениями перемещая либо камеру, либо плоскость.
3. Обнаружить особые точки на изображении.
4. Определить пять внутренних параметров и все внешние параметры, используя метод близких форм.
5. Определить коэффициенты радиального искажения с помощью метода линейных наименьших квадратов.
6. Уточнение параметров минимизацией [5].

Структурированная подсветка. Задача технического обеспечения структурной подсветки – получение пар изображений реальной комплексной сцены, где каждый пиксель имеет уникальный ярлык, которому соответствует пиксель с таким же ярлыком на другом изображении. Традиционные подходы полагались или на ручное размещение отметок (ярлыков) на небольшом количестве изображений с простой конструкцией сцены. Так же использовались смоделированные на компьютере изображения.

Техника структурированной подсветки основывается на проецировании одного или нескольких световых объектов в сцену, обычно, расположенных в определенном порядке, чтобы получить карту глубины сцены, обычно используется одна камера и один проектор. Процесс получения стереоизображений состоит из следующих этапов:

1. Получение всех желаемых образов со всеми видами подсветки (естественное освещение, от одного или нескольких проекторов).
2. Исправление изображения для получения обычной эпиполярной геометрии, используя небольшой набор соответственных объектов или плотных 2D соответствий.
3. Расшифровка световых объектов для выделения (u, v) кодов для каждого пикселя в каждом образе.

4. Вычисление соответствий, используя уникальные коды для каждого пикселя. Результат процесса поиска соответствий – вид несогласованностей.

5. Определение матрицы проекций для источников освещения из образа несогласованностей и кодов ярлыков.

6. Репроецирование ярлыков в двух видовую геометрию. Результатом является подсветка несогласованностей.

7. Объединение несогласованностей из разных источников, чтобы получить конечную, надежную и точную карту несогласованностей.

8. (Необязательный этап). Обрезка и субдискретизация карт несогласованностей и образов, полученных под естественным освещением [6].

Описание алгоритма стереообработки, реализованного в программе. Считается что все предположения., кроме последнего, верны (алгоритм игнорирует работу с заслоненными пикселями).

Алгоритм, реализованный в программе, может быть описан как (в соответствии с классификацией, представленной в [2] (page 15)):

1. Стоимости соответствия (matching cost) - абсолютная разность интенсивностей (AD) при заданном диспаритете(disparity);

2. Агрегация (комбинация) осуществляется путем суммирования стоимостей соответствия поверх квадратных окон с постоянным диспаритетом;

3. Диспаритет вычисляется путем выбора минимального (выигравшего) агрегированного значения в каждом пикселе (WTA, winner-take-all).

Данный алгоритм как результат своей работы вычисляет функцию расхождения (disparity function) по отношению к одному из изображений. Функция расхождений позволяет получить карту глубины.

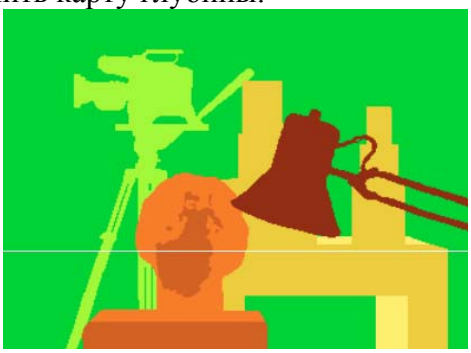


Рисунок 3 - Карта глубины для тестового изображения Tsukuba

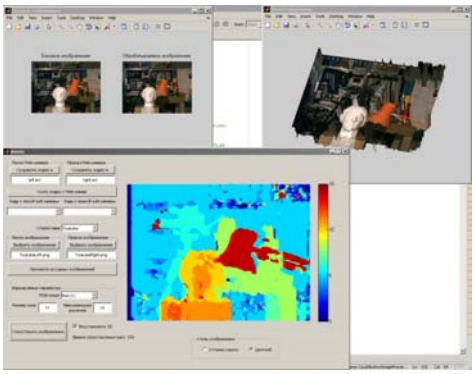


Рисунок 4

Карта глубины подразумевает низкую точность реконструкции сцены, так как сцена лишь разделяется на слои по глубине.

Характеристика алгоритма и возможности улучшения. Данный алгоритм относится к локальным алгоритмам и, как следствие, имеет более низкое качество и более высокую производительность по сравнению с глобальными алгоритмами [3,4,7,8]. Одномерные алгоритмы (к которым относится и реализованный алгоритм) обладают возможностью быть распараллеленными (в идеале: по строке на вычислительное устройство), что является большим плюсом в условиях наиболее популярного метода наращивания мощностей вычислительной техники.

Качество работы алгоритмов можно улучшить, если известна информация о сегментации изображения. Так соседним пикселям из одного сегмента, скорее всего, должны быть присвоены похожие расхождения.

Низкая производительность не позволяет использовать в лоб алгоритмы стереозрения для получения “дальностных” данных из видеоряда. Интересное направление исследования —

изучение возможностей использования связки из алгоритма сопоставления изображений и детектора движущихся объектов. Также возможно использование какого-нибудь алгоритма предсказания нахождения объекта в следующий момент. Во-первых, это сократит область изображения для вычисления новых расхождений (вычисляем новые расхождения только на движущихся объектах). Во-вторых, если используемый алгоритм итеративный (реализованный алгоритм не является итеративным), то он улучшит текущую конфигурацию. Таким образом, возможно, на вход алгоритма можно подавать предыдущую конфигурацию или предсказание конфигурации, что должно положительно сказаться на производительности.

Графический интерфейс (рисунок 4) пользователя (GUI) включает в себя 3 окна:

1. Окно "stereo" - главное окно программы, позволяющее выбирать исходные данные, формат выходных данных и параметры алгоритма стерео сопоставления.
2. Окно "Текущие изображения" – в этом окне отображается пара исходных изображений. Окно используется для предварительного просмотра изображений.
3. Окно "3D восстановление" – в данном окне отображается результат 3D восстановления сцены.

Литература

1. Richard Szeliski, Computer Vision: Algorithms and Applications (August 18, 2010 draft) // URL: http://szeliski.org/Book/
2. Scharstein D., Szeliski R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms // Int. Journal of Computer Vision 47. April-June 2002. PP. 7–42.
3. Bobick A. F., Intille S. S. Large occlusion stereo // Int. Journal of Computer Vision. 33(3). 1999. PP. 181–200.
4. Sun J., Shum H., Zheng .N Stereo matching using belief propagation // In ECCV. 2002. PP. 510–524.
5. Zhengyou Zhang. A flexible new technique for camera calibration. Technical Report MSR-TR-98-71. Aug. 13, 2008.
6. Daniel Scharstein, Richard Szeliski. High-Accuracy Stereo Depth Maps Using Structured Light. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003), volume 1, pages 195–202, Madison, WI, June 2003.
7. Kolmogorov V., Zabih R. Computing visual correspondence with occlusions using graph cuts // ICCV. Vol. 2. 2001. PP. 508–515.
8. Boykov Y., Veksler O., Zabih R. Fast approximate energy minimization via graph cuts // IEEE TPAMI 23(11). 2001. PP. 1222–1239.
9. Sun J., Shum H., Zheng .N Stereo matching using belief propagation // In ECCV. 2002. PP. 510–524.
10. Middlebury Stereo Vision Page. Evaluation. // URL: http://vision.middlebury.edu/stereo/

СИСТЕМА ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ ПРИ ВЫБОРЕ ПАРАЛЛЕЛЬНОГО АППАРАТНО-ПРОГРАММНОГО КОМПЛЕКСА ДЛЯ ПОСТРОЕНИЯ ОБЛАСТЕЙ ДОСТИЖИМОСТИ ЛЕТАТЕЛЬНОГО АППАРАТА

Федин В.А.
Научный руководитель: д.ф.-м.н., профессор Карпенко А.П.
МГТУ им. Баумана, кафедра РК6, Москва, Россия

EXPERT SUPPORT SYSTEM FOR HARDWARE AND SOFTWARE BUNDLE SELECTION FOR AIRCRAFT ATTAINABILITY DOMAINS CONSTRUNCTION

V.A. Fedin
Scientific adviser: professor A.P. Karpenko
BMSTU, Moscow, Russia

Аннотация
В работе обсуждаются основные принципы разработки системы поддержки принятия решений, предназначенной для оптимального проектирования аппаратно-программного комплекса, обеспечивающего параллельное построение границ области достижимости летательного аппарата с заданной точностью и за заданное время.

Abstract
In this paper underlying principles of expert support system development are discussed. This system is intended for optimal hardware and software bundle design, which implements parallel aircraft attainability domain boundaries construction with given accuracy and time.

Введение

Для приближенного построения области достижимости летательного аппарата (ОДЛА) могут быть использованы методы на основе многократного интегрирования исходной модельной системы обыкновенных дифференциальных уравнений (ОДУ), близкие методы, использующие полиномиальную или нейросетевую аппроксимацию правых частей этих уравнений, методы приближенного построения аппроксимаций границ ОДЛА и другие методы [1].

Указанные методы могут быть реализованы на многопроцессорных вычислительных системах с общей или распределенной памятью, системах на основе графических процессорных устройств (ГПУ), а также на нейросетевых ускорителях (НСВС) [2,3]. При этом каждому из методов может быть поставлено в соответствие несколько алгоритмов, каждый из которых может быть несколькими способами отображен на архитектуру каждой из рассматриваемых вычислительных систем. В силу многообразия вариантов состава целевого аппаратно-программного комплекса, реализующего построение ОДЛА на борту летального аппарата, возникает задача разработки программной системы для автоматизированного синтеза этого комплекса. В работе данная система рассматривается, как система поддержки принятия решений (СППР).

Полагается, что СППР функционирует на инструментальном гетерогенном вычислительном комплексе, который включает в себя все указанные классы вычислительных систем. Это дает возможность корректной оценки времени решения задачи, а также позволяет выполнять на инструментальном комплексе отладку целевых программ.

Постановка задачи

Рассмотрим динамическую систему

$$\dot{X} = F(t, X, U),\ X(0) = X^0,\ t \in [0, T], \qquad (1)$$

где $X = X(t)$ – n-мерный вектор фазовых переменных системы, $U = U(t)$ – m-мерный вектор управлений, X^0 – n-мерный вектор начальных условий, $t \in [0,T]$, $F(t,X,U)$ – n-мерная вектор-функция. Задано множество допустимых управлений D_U, так что $U \in D_U \subset L_U[0,T]$, где $L_U[0,T]$ - некоторое пространство m-мерных функций, определенных на интервале $[0,T]$, например, пространство кусочно-постоянных функций [4].

Среди компонентов вектора фазовых переменных $X = (x_1, x_2, ..., x_n)$ выделим $\nu \le n$ переменных. Не ограничивая общности, положим, что эти переменные образуют ν-мерный вектор $Y = (x_1, x_2, ..., x_\nu)^T$.

Областью достижимости $D = D(T, X^0)$ системы (1) назовем множество всех возможных значений вектора $Y(T)$, которые принимаются на решениях системы (1) при начальных условиях X^0 и выполнении условия $U \in D_U$.

Ставится задача приближенного построения области достижимости D.

Для решения поставленной задачи, очевидно, достаточно построить границу соответствующей области достижимости. В некоторых случаях могут быть известны множества управлений $D_U^\Gamma \in L_U^\Gamma[0,T] \subseteq L_U[0,T]$, которые приводят систему (1) на границу Γ области достижимости D [4]. В этом случае задача сводятся к построению границы Γ. Здесь $L_U^\Gamma[0,T]$ - некоторое подпространство функционального пространства $L_U[0,T]$.

Методы решения задачи

Подробно рассматриваемые методы приближенного построения границ ОДЛА рассмотрены в наших работах [1 - 3]. Основным таким методом является метод мультифиниша, в котором естественным образом выделяется три следующих этапа.

Этап 1. Построение функции $\widetilde{F}(t,X,U)$, аппроксимирующей функцию $F(t,X,U)$.

На этом этапе может быть реализована полиномиальная МНК-аппроксимация функции $F(t,X,U)$ на вычислительных кластерах и ГПУ, или нейросетевая аппроксимация – на указанных вычислительных системах, а также на НСВС.

Этап 2. Многократное интегрирование системы (1) и построение дискретной аппроксимации Γ_d границы области достижимости Γ с использованием функции $\widetilde{F}(t,X,U)$ вместо функции $F(t,X,U)$.

Для решения задач этого этапа рассматривается использование многопроцессорной вычислительной системы с распределенной памятью (вычислительного кластера), ГПУ с архитектурой CUDA, а также НСВС.

Схемы распараллеливания вычислений и особенности отображения метода мультифиниша на архитектуры вычислительного кластера и ГПУ рассмотрены в наших работах [2, 3]. В этих же работах приведены соответствующие результаты исследования эффективности распараллеливания.

Этап 3. Построение непрерывной аппроксимации Γ_c границы ОДЛА.

Построение границы Γ_c также может быть выполнено на вычислительных кластерах, ГПУ и НСВС. Использование с этой целью последних вычислителей рассмотрено в наших работах [5, 6].

Основные функции системы поддержки принятия решений

Обозначим $\mathbf{M}_1 = \{M_{1,i}\}$ - рассматриваемую совокупность методов аппроксимации функции $F(t,X,U)$. Аналогичные обозначения $\mathbf{M}_2$, $\mathbf{M}_3$ введем для совокупностей

рассматриваемых методов построения границ Γ_d, Γ_c соответственно. Здесь полагается, что методы $\mathbf{M}_2$ включают в себя также и методы интегрирования систем ОДУ (1).

Таким образом, для приближенного построения границ ОДЛА может быть использована иерархическая совокупность численных методов вида

$\mathbf{M}_1 \rightarrow \mathbf{M}_2 \rightarrow \mathbf{M}_3$.

Каждому из методов $\mathbf{M}_1$, $\mathbf{M}_2$, $\mathbf{M}_3$ можно поставить в соответствие некоторую совокупность алгоритмов $\mathbf{A}_{i,j} = \{A_{i,j,k}(\mathbf{P}_{i,j,k})\}$, где $A_{i,j,k}$ - *k*-й алгоритм, реализующий метод $M_{i,j}$, а $\mathbf{P}_{i,j,k}$ - вектор свободных параметров этого алгоритма. В случае алгоритма $A_{1,j,k}$, например, этот вектор может включать в себя порядок аппроксимирующего полинома; в случае алгоритма $A_{2,j,k}$ - число узлов сетки, покрывающей множество D_U; в случае алгоритма $A_{3,j,k}$ - также порядок аппроксимирующего полинома.

Назовем проектируемую бортовую вычислительную систему «целевой», а вычислительные системы, на которых функционирует рассматриваемая СППР, «инструментальными». В качестве вариантов реализации целевой ЭВМ рассмотрим следующие классы систем:

- вычислительный кластер $\mathbf{C}_1 = \{C_1(\mathbf{P}_1)\}$;
- ГПУ $\mathbf{C}_2 = \{C_2(\mathbf{P}_2)\}$;
- НСВС $\mathbf{C}_3 = \{C_3(\mathbf{P}_3)\}$

Здесь $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$ - векторы свободных параметров соответствующих ЭВМ (например, число процессоров для вычислительного кластера, число мультипроцессоров для ГПУ, число нейрочипов для НСВС).

Множество вариантов отображения алгоритма $A_{i,j,k}(\mathbf{P}_{i,j,k})$ на архитектуру системы $C_l(\mathbf{P}_l)$ обозначим

$\mathbf{E}(A_{i,j,k}(\mathbf{P}_{i,j,k}), C_l(\mathbf{P}_l)) = \{E_m(A_{i,j,k}(\mathbf{P}_{i,j,k}), C_l(\mathbf{P}_l))\}$.

Задача оптимального проектирования аппаратно-программного комплекса, обеспечивающего построение ОДЛА за заданное время с заданной точностью, состоит в многокритериальном выборе методов M_{1,i^*}, M_{2,j^*}, M_{3,k^*}, соответствующих алгоритмов $A_{1,i^*,l^*}(\mathbf{P}^*_{1,i,j})$, $A_{2,k^*,l^*}(\mathbf{P}^*_{2,k,l})$, $A_{3,m^*,n^*}(\mathbf{P}^*_{3,m,n})$, целевой ЭВМ $C_{q^*}(\mathbf{P}^*_q)$ и, наконец, в выборе вариантов отображения указанных алгоритмов на архитектуру выбранной ЭВМ.

Процесс решения задачи оптимального проектирования является итерационным и включает в себя несколько вложенных циклов:

– выбор методов $M_{1,i}$, $M_{2,j}$, $M_{3,k}$;

– выбор соответствующих алгоритмов $A_{1,i,l}$, $A_{2,j,m}$, $A_{3,k,n}$;

– выбор значений параметров $\mathbf{P}_{1,i,l}$, $\mathbf{P}_{2,j,m}$, $\mathbf{P}_{3,k,n}$ этих алгоритмов;

– выбор класса целевой вычислительной системы $\mathbf{C}_q$;

– выбор значений вектора параметров $\mathbf{P}_q$ и, тем самым, целевой системы $C_q(\mathbf{P}_q)$;

– выбор вариантов отображения алгоритмов $A_{1,i,l}(\mathbf{P}_{1,i,l})$, $A_{2,j,m}(\mathbf{P}_{2,j,m})$, $A_{3,k,n}(\mathbf{P}_{3,k,n})$ на вычислительную систему $C_q(\mathbf{P}_q)$;

– построение аппроксимации границы Γ_c ОДЛА на инструментальной вычислительной системе класса $\mathbf{C}_q$ с помощью выбранных методов, алгоритмов и их параметров;

– оценка точности полученной аппроксимации;

– в случае неудовлетворительной точности, последовательные попытки изменить значения параметров алгоритмов, выбрать другой алгоритм и, наконец, - другой метод;

– оценка времени решения задачи на системе $C_q(\mathbf{P}_q)$;

– в случае неудовлетворительного времени решения задачи, последовательные попытки изменить значения параметров системы и класса системы.

Разрабатываемая СППР призвана облегчить лицу, принимающему решения (ЛПР), или группе лиц, принимающих решения (ГПР), решение указанной задачи оптимального проектирования. Основной задачей, которую должна решать эта СППР, является многокритериальная оценка вариантов параллельного аппаратно-программного комплекса. В качестве критериев качества решения при этом могут использоваться вес вычислительной системы, ее габариты, энергопотребление, стоимость и т.д.

Заключение

В работе изложены основные принципы построения системы поддержки принятия решений, предназначенной для оптимального проектирования аппаратно-программного комплекса, обеспечивающего параллельное построение границ области достижимости летательного аппарата с заданной точностью и за заданное время.

В качестве целевой вычислительной системы предполагается использование кластерной системы, системы на основе графических процессорных устройствах или системы на основе и нейросетевых ускорителей. Инструментальная вычислительная система полагается гетерогенной, включающей в себя указанные типы параллельных вычислительных систем.

Рассмотрена иерархическая совокупность численных методов приближенного построения границ ОДЛА, включающая в себя различные методы аппроксимации векторного поля модельной системы обыкновенных дифференциальных уравнений, методы построения дискретной аппроксимации границы ОДЛА, а также методы построения непрерывной аппроксимации этой границы.

В настоящее время ведутся работы по реализации рассматриваемой системы поддержки принятия решений. Авторы выражают благодарность Воронову Е.М. и Карпунину А.А. за постановку задачи и многочисленные плодотворные обсуждения подходов к ее решению.

Литература

1. Воронов Е.М., Карпенко А.П., Козлова О.Г., Федин В.А. Численные методы построения области достижимости динамической системы // Вестник МГТУ. Сер. “Приборостроение”. 2010. №2(79). С. 3-19.
2. Воронов Е. М., Карпенко А. П., Федин В. А. Параллельное построение множества достижимости высокоманевренного летательного аппарата методом "мультифиниша" // Параллельные вычислительные технологии (ПаВТ’2010): Труды международной научной конференции / ЮУрГУ. Челябинск: ЮУрГУ, 2010. С. 113-120.
3. Витюков Ф.А., Домашнев В.К., Карпенко А.П., Федин В.А. Построение области достижимости динамической системы на NVIDIA и AMD графических процессорах // Научный сервис в сети Интернет: суперкомпьютерные центры и задачи 2010: Труды международной суперкомпьютерной конференции / МГУ. Москва: МГУ, 2010. С. 635-641.
4. Воронов, Е.М., Карпунин А.А. Алгоритм оценки границ области достижимости летательного аппарата с учетом тяги // Вестник МГТУ. Сер. Приборостроение. 2007. №4(69). С. 81-99.
5. Козлова О.Г. Нейросетевая аппроксимация границы области достижимости летательного аппарата. 1. Двухмерный случай // Наука и образование: электронное научно-техническое издание. 2009. №7.
URL http://technomag.edu.ru/doc/129990.html (дата обращения: 11.12.2010).
6. Козлова О.Г. Нейросетевая аппроксимация границы области достижимости летательного аппарата. 2. Трехмерный случай // Наука и образование: электронное научно-техническое издание. 2009. №8.
7. URL http://technomag.edu.ru/doc/130282.html (дата обращения: 11.12.2010).

ЭФФЕКТИВНОСТЬ ПОСТРОЕНИЯ ФРОНТА ПАРЕТО МЕТОДОМ РОЯ ЧАСТИЦ

Хасанова Р.В.
Научный руководитель д. ф.-м. н. профессор кафедры САПР Карпенко А.П.
МГТУ им. Н.Э. Баумана, Москва, Россия

INVESTIGATION OF PARTICLE SWARM OPTIMIZATION EFFICIENCY IN APPROXIMATE CONSTRUCTION OF THE PARETO FRONT

R.V. Khasanova
D. Sc., prof. A.P.Karpenko
MSTU named after Bauman, Moscow, Russia

Аннотация
Рассматривается однороевый и многороевый методы роя частиц для приближенного построения множества Парето в задаче многокритериальной оптимизации. Приводятся постановка задачи, описание используемых алгоритмов и реализующего их программного обеспечения, а также результаты оценки их эффективности.
Abstract
Single- and multi-swarm particle swarm methods for approximate construction of the Pareto set in multi-criteria optimization problem are considered. Problem statement, description of the algorithms and relevant software used, and effectiveness estimates are reviewed.

В настоящее время при решении задач оптимизации все более широкое распространение получают стохастические поведенческие методы [1]. Одним из таких методов является метод роя частиц (*Particle Swarm Optimization*, *PSO*), основанный на закономерностях социального поведения [2-3]. Достаточно новым является применение метода *PSO* в задаче многокритериальной оптимизации (*Multi-Objective Swarm Optimization*, *MOPSO*) [1]. В работе рассматривается применение этого метода для приближенного построения множества Парето в указанной задаче.

Постановка задачи. Совокупность частных критериев оптимальности $\Phi_k(X), k \in [1:s]$ образует векторный критерием оптимальности $\Phi(\mathrm{X})$, где X - n-мерный вектор варьируемых параметров. Положим, что ставится задача минимизации каждого из частных критериев в одной и той же области допустимых значений $D_X \in R^n$:

$$\min_{\mathrm{X}\in \mathrm{D_X}} \Phi(\mathrm{X}) = \Phi(\mathrm{X}^*), \tag{1}$$

где D_X - множество допустимых значений вектора варьируемых параметров; R^n - n-мерное арифметическое пространство.

Векторный критерий оптимальности $\Phi(X)$ выполняет отображение множества допустимых значений $D_{\mathrm{X}} \subset \{\mathrm{X}\}$ в некоторую область $D_{\Phi} \subset \{\Phi\}$, где $\{\mathrm{X}\}$ — пространство варьируемых параметров. Выделим из множества D_{Φ} подмножество $D_{\Phi}^* \in D_{\Phi}$ точек, для которых нет точек, их доминирующих. Это множество называется фронтом Парето. Множество $D_X^* \in D_X$, соответствующее множеству D_{Φ}^*, называется множеством Парето (переговорным множеством, областью компромисса) [3].

Ставится задача приближенного построения множества Парето (а, тем самым, и фронта Парето) в задаче многокритериальной оптимизации (1).

Канонический метод роя частиц (PSO). Множество частиц обозначим $\mathbf{P} = \{P_i, i \in [1:N]\}$, где N – число частиц в рое (размер популяции). В дискретный момент

времени $t=[0:T]$ координаты частицы P_i определяются вектором $X_{i,t}=(x_{i,t,1},x_{i,t,2},...,x_{i,t,n})$, а ее скорость - вектором $V_{i,t}=(v_{i,t,1},v_{i,t,2},...,v_{i,t,n})$. Здесь T - число итераций. Начальные координаты и скорости частицы P_i равны $X_{i,0}=X_i^0$, $V_{i,0}=V_i^0$ соответственно, где X_i^0 - матрица случайных чисел, V_i^0 - нулевая матрица.

Итерации в каноническом методе PSO выполняются по схеме

$$V_{i,t+1}=\alpha V_{i,t}+U_1[0,\beta]\otimes\left(X_{i,t}^b-X_{i,t}\right)+U_2[0,\gamma]\otimes\left(X_{\mathrm{g},t}-X_{i,t}\right), \quad (2)$$

$$X_{i,t+1}=X_{i,t}+V_{i,t+1}. \quad (3)$$

Здесь $U_l[a,b],l=1,2$ представляет собой n-мерный вектор псевдослучайных чисел, равномерно распределенных в интервале $[a,b]$; $\otimes$ - символ покомпонентного умножения векторов; $X_{i,t}^b$ - вектор координат частицы P_i с наилучшим значением целевой функции $\Phi(\mathrm{X})$ за все время поиска $[0:t]$; $X_{g,t}$ - вектор координат соседней с данной частицы с наилучшим за время поиска $[0:t]$ значением целевой функции $\Phi(\mathrm{X})$; α,β,γ - свободные параметры алгоритма. Важнейшее в методе *PSO* понятие соседства частиц зависит от используемой топологии соседства и определено, например, в работе [2].

Однороевый метод многокритериальной оптимизации. Важной частью метода *MOPSO* является определение глобально лучшей частицы (в смысле формулы 1) для каждого индивидуума в популяции. В многокритериальной задаче глобально лучшую частицу следует искать на множестве Парето. С этой целью в данном методе используется архив частиц A. В нем хранятся недоминируемые друг другом координаты частиц $X_{i,t},t=1,2,...,T$.

В результате работы алгоритма *MOPSO* на итерации t происходит обновление архива A_t. Сравнение частиц текущего поколения из архива A_t с множеством частиц на итерации *t* выполняет функция *Update*. Если некоторая частица текущего поколения P_i доминирует частицу P_j из архива, то доминирующая частица должна заменить частицу архива. Если P_j доминирует P_i, то частицу текущего поколения не нужно добавлять в архив. Если же никакая из частиц не является доминирующей, то частица P_i добавляется к частицам архива. Таким образом, на первой итерации, когда архив пуст, функция *Update* добавляет в архив все частицы текущего поколения, которые не доминируют друг друга.

Выбор глобально лучшей частицы осуществляет функция *FindGlobalBest*. Сначала в функции вычисляется евклидово расстояние от частицы до других частиц P_i архива A_t. Наилучшая для частицы P_i будет та частица P_j, расстояние до которой наименьшее.

Итерации могут продолжаться до тех пор, пока множество недоминируемых решений не перестанет меняться, либо до достижения заданного числа итераций.

Многороевый метод многокритериальной оптимизации. Миграция частиц между роями осуществляется через фиксированное число итераций Δ. Для каждого из роев S_i поддерживается архив недоминируемых частиц A^i. Во время миграции обновляется архив недоминируемых частиц, которые были получены за все время поиска всеми роями. Обозначим его A^S. Далее каждый рой вновь осуществляет поиск недоминируемых решений, но при этом на первой итерации после миграции архив каждого из роев S_i не пуст, а $A^i=A^S$. Число миграций является свободным параметром метода.

Начальные положения частиц выбираются по следующей схеме. Область допустимых значений D_X разбивается на подобласти равного объема, число которых совпадает с числом подроев. Для частиц из подроя P^i начальные положения частиц выбираются случайным

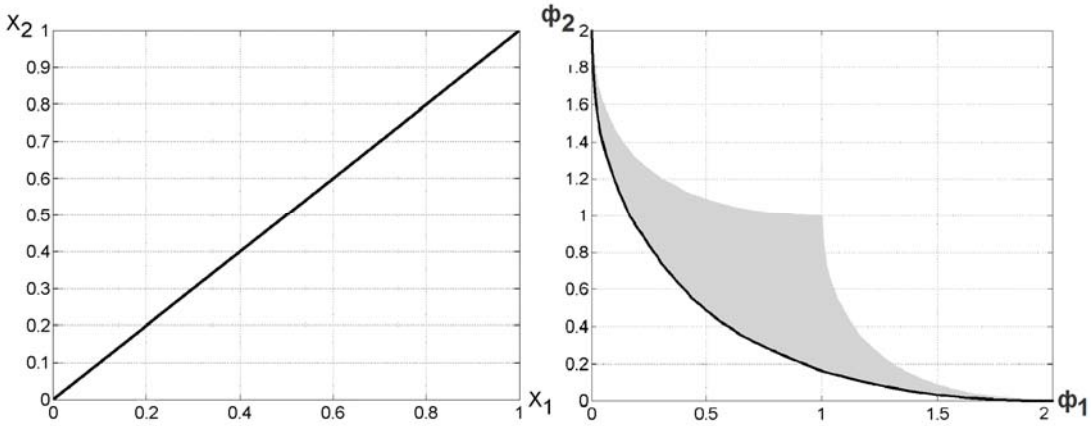
образом в *i*-ой подобласти. Для того, чтобы частицы не выходили за пределы свой подобласти, используется метод штрафных функций.

Исследование эффективности метода *MOPSO*. Исследование выполнено для тестовой задачи

$$\begin{cases} \phi_1(X) = x_1^2 + x_2^2, \\ \phi_2(X) = (x_1 - 1)^2 + (x_2 - 1)^2. \end{cases} \tag{4}$$

$$D_X = \{X \mid 0 \le x_i \le 1, i \in [1,2]\}. \tag{5}$$

Известно, что множество Парето и фронт Парето для этой задачи имеют вид, представленный на рисунке 1.



а) б)

Рисунок 1 - Множество Парето (а) и фронт Парето (б) задачи (4), (5)

Исследования эффективности одноroевого метода *MOPSO* выполнено при числе частиц в рое, равном 10 и при числе итераций, равном 100. Приближенные фронт и множество Парето, полученные одноroевым методом *MOPSO*, представлены на рисунке 2. Рисунок соответствуют ситуации, когда архив недоминируемых частиц содержит 315 частиц.

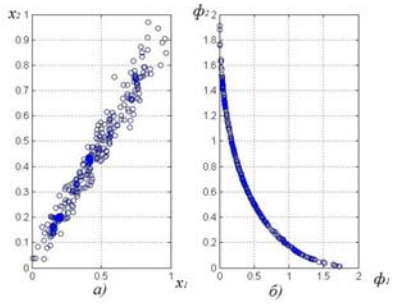


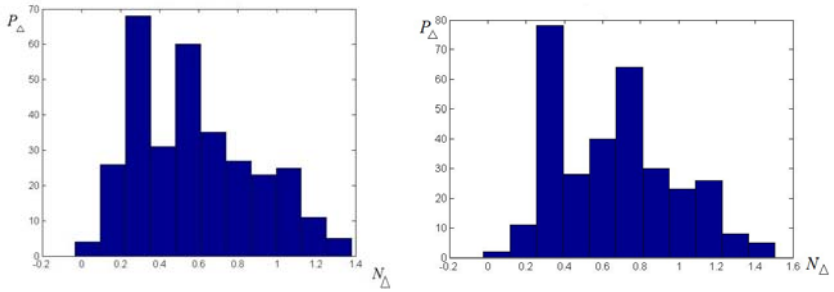
Рисунок 2 – Приближенные множество Парето (а) и фронт Парето (б): одноroевый метод

Рисунок 2 показывает, что одноroевый метод *MOPSO* обеспечивает более плотное покрытие фронта и множества Парето в центральных областях этих множеств и менее плотное – в периферийных областях (рисунок 3).

Исследования эффективности многороевого метода выполнено при следующих значениях параметров: числе частиц в подрое - 10; число подроев - 100; период миграции частиц между подроями - 1; число итераций - 100. Соответствующие приближенные фронт и множество

Парето представлены на рисунке 4. Архив недоминируемых частиц содержит в данном случае 456 частиц. Соответствующие гистограммы приведены на рисунке 5.

Рисунки 4, 5 показывают существенно более высокую равномерность покрытия множества и фронта Парето по сравнению с одноroевым методом.



а) б)

Рисунок 3 - Плотность покрытия множества Парето (а) и фронта Парето (б): одноroевый метод

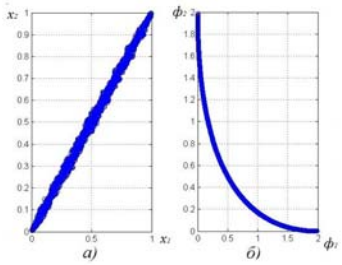
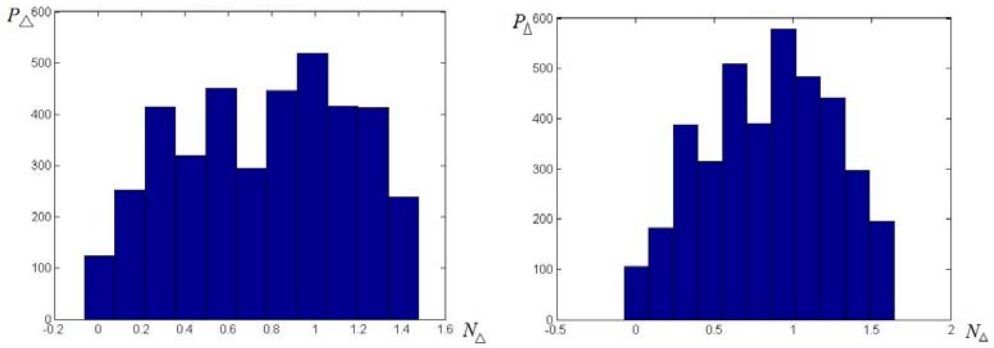


Рисунок 4 - Множество Парето (а) и фронт Парето (б): многороевый метод



а) б)

Рисунок 5 - Плотность покрытия множества Парето (а) и фронта Парето (б): многороевый метод

Литература

1. Mostaghim S., Teich, J. Strategies for Finding Good Local Guides in Multi-Objective Particle Swarm Optimization (MOPSO) // Swarm Intelligence Symposium: Proceeding, 2003. - pp. 26–33.
2. Карпенко А.П., Селиверстов Е.Ю. Глобальная оптимизация методом роя частиц. Обзор // Информационные технологии, 2010, № 2, С. 25-34.
3. Субботин С.А., Олейник Ан.А., Олейник Ал.А. PSO-метод // Интеллектуальные мультиагентные методы (Swarm Intelligence), 2006, №3, С. 55-70.
4. Карпенко А.П. Методы оптимизации [Электронный ресурс].- (http://bigor.bmstu.ru).

ПОСТРОЕНИЕ СТАТИЧЕСКОГО 3D ИЗОБРАЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ КОРРЕЛЯЦИОННОГО МЕТОДА

Шмакова Н.А.
Научный руководитель: к.т.н., доцент Волосатова Т.М.
Кафедра САПР (РК6) МГТУ им. Н.Э. Баумана, Москва, Россия

A STATIC 3D IMAGE CREATING USING CORRELATION METHOD

Shmakova N.A.
The scientific chief: Ph.D., assistant professor Volosatova T.M.
CAD/CAE Department (RK6), BMSTU, Moscow, Russia

Аннотация
В докладе рассмотрена задача построения статического 3D изображения по стереопаре, ее геометрическая интерпретация и корреляционный метод решения. Приведено описание программного средства, разработанного на базе матричной системы компьютерной математики MATLAB, которое распознает по двум снимкам объекта его рельеф при помощи встроенной корреляционной функции. Также представлен пример работы программы для стереопары, полученной по параллельной схеме с известным фокусным расстоянием и базой.

Abstract
This article deals with the task of creating a static 3D image using a stereo pair, its geometrical interpretation and correlation method of the solution. The software, developed on the base of a matrix system of computer mathematic MATLAB, was described. It recognizes the object's relief using its two images and built-in correlation function. The program output for stereo pair, which was created using parallel scheme with known focal length and base, was included into article.

Реальные объекты сцены находятся в трехмерном пространстве. Для наблюдения за ними большинство биологических объектов, в том числе и человек, имеет два глаза, разнесённых примерно на 65 мм. Благодаря этому мы легко вычисляем расстояния до предметов и, как следствие, определяем с высокой точностью их истинные размеры.

Аналогичная цель стоит перед системами технического зрения (СТЗ). Они предназначены для решения задачи по дополнению или даже замене человека в областях деятельности, связанных со сбором и анализом зрительной информации [4].

Существуют две основные схемы получения стереоскопического изображения: параллельная и перекрестная, моделирующая конвергенцию глаз человека. В первом случае оси камер параллельны, что упрощает процесс съемки. Поэтому параллельная схема применяется чаще. Во втором случае оси камер пересекаются. В результате получается более обширная зона наложения изображений, и вычисление расстояний до объектов не требует предварительного компенсационного сдвига как в первом случае.

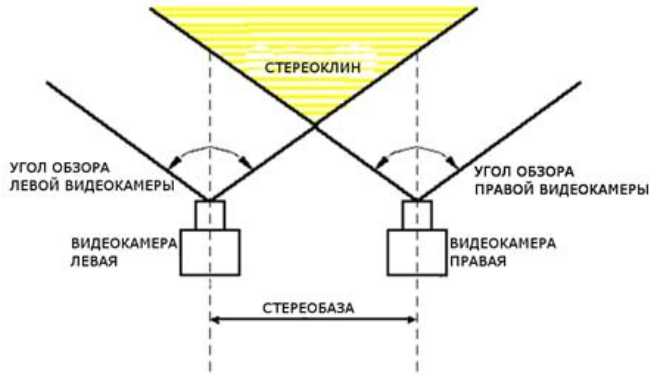


Рисунок 1 - Схема получения стереоскопического изображения

Общая схема искусственного получения стереоскопических изображений приведена на рисунке 1, где две видеокамеры разнесены на некоторую базу. Каждая видеокамера имеет свой угол обзора, а пересечение углов обзора левой и правой видеокамер образует стереоклин. Определение расстояния возможно только в области стереоклина.

Решение задачи построения 3D изображения включает два основных этапа:

- Нахождение соответствующих точек на изображениях стереопары;
- Вычисление объемных координат объектов, изображенных на стереопаре.

Общий принцип определения пространственного расположения точек может быть рассмотрен на примере параллельной схемы получения стереоскопического изображения – рисунок 2.

Оси камер *X* коллинеарны, а оси *Y* и *Z* параллельны. Ось *Y* перпендикулярна плоскости рисунка. Начало координат, или центр проектирования правой камеры, смещен относительно центра левой камеры вдоль оси *X* на величину *b,* которая называется базой (baseline) стереоскопической системы. При наблюдении некоторой точки объекта *P* на левом изображении формируется точка *Pl*, а на правом — точка *Pr*. Из геометрических построений очевидно, что точка *P* лежит на пересечении лучей *LPl* и *RPr*.

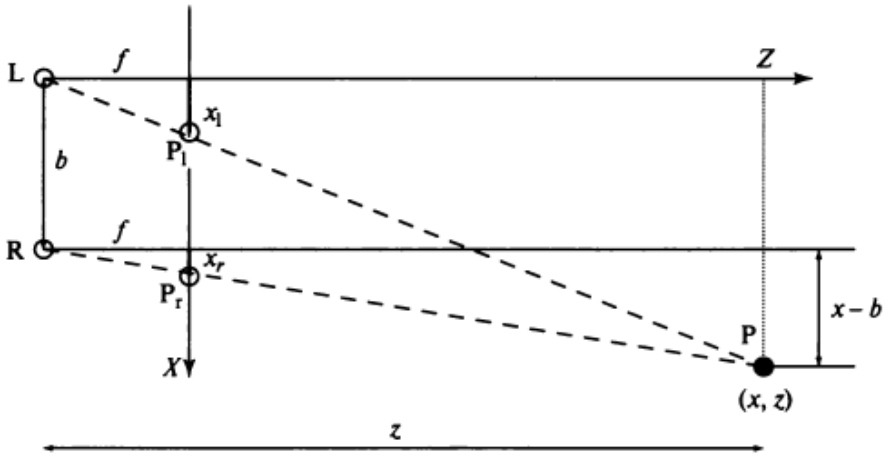


Рисунок 2 - Геометрическая модель простой стереоскопической системы

В качестве общей системы координат выбрана система координат левой камеры *L*. Все координаты измеряются относительно базиса левой камеры *L*, за исключением x_r, которая измеряется относительно системы координат правой камеры *R*.

В результате, получим уравнения (1) [4]:

$$z = fb/(x_l - x_r) = fb/d$$

$$x = x_l z / f = b + x_r z/f$$

$$y = y_l z / f = y_r z/ f$$

В выражение глубины точки *P* введена величина рассогласования *d.* Она равна разности координат x_l и x_r точек на левом и правом изображениях. Решение уравнений (1) дает все три координаты, полностью определяющие положение точки *P* в трехмерном пространстве. Из уравнений (1) очевидно, что расстояние до точки *P* увеличивается при уменьшении рассогласования и уменьшается при увеличении рассогласования. Расстояние стремится к бесконечности при приближении рассогласования к нулю.

При наблюдении реальных трехмерных сцен поиск соответствующих точек часто оказывается весьма сложной задачей, так как среди большого количества точек поверхностей может быть неясно, какие точки левого и правого изображений являются образами одной и той же точки трехмерной поверхности. Как правило, камеры в стереоскопических системах очень точно выравниваются так, чтобы ограничить область поиска соответствующих точек двух изображений в пределах строк с одинаковыми индексами.

Хотя для стереосистем известно и используется много различных ограничений, все еще остаются нерешенные проблемы. Одна из проблем возникает в случае, когда точка *P* видна только на одном изображении. Противоположная по смыслу проблема часто возникает при наличии слишком малого количества признаков точек. Эта проблема характерна для сцен с гладкими объектами без текстуры (например, таких, как покрытый снегом холм).

Наиболее сложные операции обработки в стереоскопических зрительных системах связаны не с вычислениями глубины, а с определением соответствующих признаков, которые необходимы для этих вычислений. Некорректно установленные соответствия приводят к ошибкам в значениях глубины. Ошибки могут приводить как к незначительным отклонениям от истинного значения, так и к полностью неверным результатам [4].

Один из самых распространенных методов обнаружения соответствия между пикселями двух изображений основан на использовании оператора кросс-корреляции. Корреляционные методы — это нахождение пиксельных соответствий путем сравнения профилей яркости в окрестности потенциально соответствующих точек разных изображений объекта [3,4]. Пиксель второго изображения, для которого достигается максимальный отклик оператора кросс-корреляции, считается наилучшим вариантом сопоставления для пикселя первого изображения и используется для вычисления глубины соответствующей пространственной точки.

Корреляционная функция двух функций *f(x,y)* и *h(x,y)* определяется следующим выражением:

$$f(x,y)\circ h(x,y)=\frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}f^{*}(m,n)h(x+m,y+n)$$

где *f** - обозначает функцию, комплексно-сопряженную функции *f*. В данном случае мы имеем дело с действительными функциями, где *f*=f*.

Имеет место теорема о корреляции. Пусть *F(u,v)* и *H(u,v)* обозначают фурье-образы функций *f(x,y)* и *h(x,y)* соответственно. Теорема констатирует, что корреляция в пространственной области *f(x,y) ◦h(x,y)* и произведение *F*(u,v)H(u,v)* образуют фурье-пару. Этот результат, который можно записать в виде

$$f^{*}(x,y)h(x,y)\Leftrightarrow F(u,v)\circ H(u,v)$$

означает, что корреляционная функция в пространственной области может быть получена как результат обратного преобразования Фурье, примененного к произведению *F*(u,v)H(u,v)*, где *F** комплексно сопряженная к *F* функция [5].

MATLAB и пакет Image Processing предоставляют широкий спектр средств для цифровой обработки и анализа изображений. Это освобождает разработчиков от выполнения длительных операций кодирования и отладки алгоритмов, позволяя сосредоточить усилия на решении основной научной или практической задачи [1,2]. Поэтому было решено использовать систему MATLAB как платформу для написания ПО.

Разработанное программное средство через пользовательский интерфейс получает все необходимые для создания статического 3D изображения данные, а затем представляет результат в виде рельефной поверхности.

Обработка входных изображений в среде MATLAB проходит в 3 этапа.

На первом этапе выполняется чтение выбранных изображений в формате jpg или bmp в трехмерный массив с RGB представлением цветов. Далее происходит преобразование к формату представления HSV посредством встроенной функции rgb2hsv() [2,6]. И на последующую обработку отправляется уже только V(value) составляющая от каждого изображения, так как ее достаточно для корректной работы метода кросс-корреляции.

Второй этап относится к определению соответствующих точек на двух изображениях с тем условием, что стереопара имеет только горизонтальный параллакс. Далее в цикле для каждого пикселя левого изображения производится поиск соответствующего пикселя правого изображения по результатам операции нормированной кросс-корреляции, которую выполняет встроенная функция normxcorr2() [2,6]. Эта функция использует быстрое

преобразование Фурье из-за чего время ее работы существенно меньше, чем время работы функции ненормированной кросс-корреляции.

На последнем этапе осуществляется вычисление объемных координат каждой точки изображения:

-если база и/или фокус не заданы, то координата *Z* принимается равной горизонтальному параллаксу,

-если база и фокус заданы, но нет угла конвергенции (или он равен нулю), то координата *Z* вычисляется по формуле (1),

-если задан угол конвергенции осей фотокамер, то используется модифицированная формула параллакса для вычисления *Z*, а именно:

$$p = f \cdot \cos(\alpha)\,[tg(\beta) - tg(\gamma)], \text{где } \beta = arctg\left(\frac{x_r}{f}\right) - \frac{\alpha}{2}, \quad \gamma = arctg\left(\frac{x_l}{f}\right) + \frac{\alpha}{2}.$$

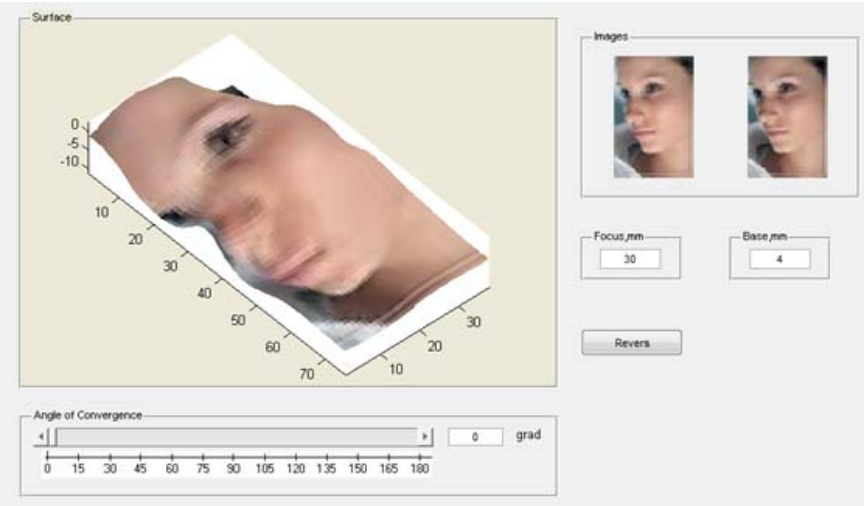


Рисунок 3 - Результат работы программы

При выводе на экран выполняется сглаживание поверхности по Z.

Результатом работы программы является поверхность, отображающая рельеф объектов, запечатленных на изображениях стереопары. Программа предоставляет пользователю возможность сохранить результаты обработки в mat-файле и, при необходимости, извлечь эти данные для просмотра. Также имеется возможность неограниченное число раз варьировать фокус, базу и угол конвергенции осей для получения наилучшего результата, если они не были заранее известны.

Литература

1. Гонсалес Р., Вудс Р., Эддинс С. Цифровая обработка изображений в среде MATLAB. Москва: Техносфера,2006.-616с. ISBN 5-94836-092-X
2. Дьяконов В.П. MATLAB. Полное руководство. – М.: ДМК Пресс, 2010. – 768с.: ил. ISBN 978-5-94074-652-2
3. Форсайт, Дэвид А., Понс, Жан. Компьютерное зрение. Современный подход.: Пер. с англ. – М.: Издательский дом «Вильямс», 2004. – 928 с.: ил. – Парал. тит. англ. IBSN 5-8459-0542-7
4. Компьютерное зрение / Л. Шапиро, Дж. Стокман; Пер. с англ. - М.: БИНОМ. Лаборатория знаний, 2006.-752 с., 8с. ил.: ил. ISBN 5-94774-384-1
5. Гонсалес Р., Вудс Р. Цифровая обработка изображений. - Москва: Техносфера, 2005.-1072с. ISBN 5-94836-028-8
6. Сайт сети Internet: Консультационный центр MATLAB компании SoftLine - http://www.nsu.ru/matlab/MatLab_RU/default.asp.htm

СИСТЕМА РАСПОЗНАВАНИЯ ДОРОЖНЫХ ЗНАКОВ

Ященко А. В.
Научный руководитель: аспирант ВМК МГУ, Чигорин А. А.
Московский Государственный Технический Университет им. Н. Э. Баумана, ИУ4

TRAFFIC SIGN DETECTION SYSTEM

Yashenko A. V.
Supervisor: P.G. Chigorin A. A.
Bauman Moscow State Technical University, Moscow, Russia

Аннотация
В статье представлена концепция системы распознавания дорожных знаков. Система реализована в среде Matlab 2008b на основе HOG [1] дескриптора и IK-SVM [2] классификатора. Представлены примеры исходных данных, дан краткий обзор используемых алгоритмов, описаны полученные результаты. В заключении сказано о практической реализации данного алгоритма.

Abstract
Traffic sign detection system is presented. System was written in Matlab 2008b environment, and based on HOG descriptor and IK-SVM classifier. Dataset examples, used algorithms, experiment results are given. Concluding, we presented information about aspects of problem practical realization of the problem.

Введение

В настоящее время накоплено огромное количество фото и видео информации, требующей анализа, в этом свете задача распознавания изображений становится особенно актуальной. В докладе рассматривается принцип действия системы распознавания дорожных знаков.

Структурная схема алгоритма представлена на (рис. 1) Для извлечения признаков из изображений был использован HOG (Histogram of oriented gradients) дескриптор [1]. В общем случае данный дескриптор не инвариантен к повороту и масштабированию, но на слабо вариативных объектах типа знаков показывает хорошие результаты. Использование данного дескриптора возможно так же для детектирования знаков, но в данной статье не будем касаться этой темы.

Для классификации объектов был использован метод опорных векторов (SVM), а именно две библиотеки: lib_svm и osu_svm, написанные на C++. Основные эксперименты проводились с библиотекой lib_svm [8], так как она поддерживает ядро intersection kernel выполненное специально для классификации гистограмм, что соответствует нашему случаю. В osu_svm использовалось линейное ядро классификатора. Описаны результаты экспериментов с обоими ядрами.

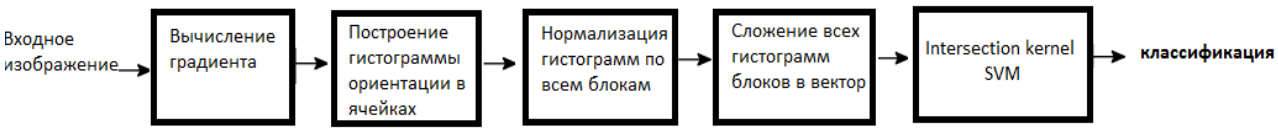


Рисунок 1 – Структурная схема алгоритма

Всего в базе представлено 43 класса знаков различной формы, (рис. 2, *а, б, в*) Фотографии знаков представляют собой поток изображений из видео. В базе присутствуют изображения очень маленького размера (рис. 2, *а*). В состав вектора признаков не были включены

цветовые компоненты, так как при построении матрицы ошибок (confusion matrix) было выявлено, что вне одной цветовой группы процент ошибки очень низок.

а

б

в

Рисунок 2 – Знаки из тренировочной выборки

При проведении экспериментов для тренировочной и тестируемой выборки были использованы две части базы знаков. В тестовой выборке содержалось 12569 изображений, а в тренировочной 25 920. Таким образом, тренировочная выборка состояла примерно из 67% изображений.

1 Гистограммы ориентированных градиентов (HOG descriptor)

1.1 Концепция распознавания и детекции объектов

Histogram of oriented gradients (HOG) – дескриптор, широко используемый в компьютерном зрении и цифровой обработке изображений для распознавания и детекции объектов. В основе данного алгоритм лежит учет ориентации направления градиента изображения в локальных областях, называемых ячейками. Впервые данный алгоритм был описан в июне 2005 года. В своих экспериментах, авторы использовали его для детекции людей. (Navneet Dalal and Bill Triggs) [2]. Алгоритм основан на том, что присутствие объекта на изображении, может быть описано с помощью гистограмм ориентации градиента изображения. Реализация алгоритма достигается за счет деления изображения на маленькие ячейки и подсчета для каждого пикселя ячейки направления градиента. Помимо маленьких ячеек существуют ещё блоки, содержащие в себе несколько ячеек. Блоки используются для нормализации всех значений градиента в ячейке по локальной окрестности. Причем блоки формируются таким образом, чтобы каждая ячейка содержится в нескольких из них (рис. 3). Нормализация дает инвариантность к освещенности. Так как дескриптор оперирует гистограммами малых ячеек, то небольшие трансформации изображения не будут вносить явные изменения в вектор признаков. Алгоритм получения конечного вектора дескриптора состоит из следующих шагов:

- Вычисление градиента изображения реализуется сверткой изображения f с маской g. Данная операция подразумевает применение маски к каждому пикселю изображения. Значение пикселя в этом случае заменяется на сумму взвешенных значений пикселей окрестности. Весовые коэффициенты определяются маской.
- Далее происходит деление изображения на ячейки. В каждой ячейки реализуется подсчет гистограммы ориентаций градиентов – каждый пиксель вносит вклад с весовым коэффициентом в гистограмму направления. При этом весь диапазон направлений (360 signed или 180 unsigned) делится на n дискретных частей. Таким образом, каждая ячейка представлена гистограммой из n столбцов.
- Следующий этап – объединение маленьких ячеек в более крупные блоки. Это объединение необходимо, чтобы локально нормализовать все значения гистограмм ячеек, входящих в блоки. При объединение необходимо учитывать, что блоки пересекаются. Таким образом, гистограмма каждой ячейки будет нормализована несколько раз относительно разных блоков.
- Конечный шаг алгоритма – получения вектора признаков дескриптора, путем объединения всех элементов гистограмм блоков.

1.2 Реализация

При реализации кода наилучший результат был получен при следующих параметрах дескриптора:

-Размер изображения: *60x60*
- Размер ячейки: *15x15*
- Весовой коэффициент: *1*
- Ориентация градиента: *signed*
- Число элементов в гистограмме n*: 9*

-Маски, применяемые при вычисление градиента: $[-1,0,1]$ *u* $[-1,0,1]^T$

-Тип нормализации: $L1-norm:\ f=\frac{v}{\|v\|_1+e}$ (1)

-Размер блоков: 2x2 ячейки

Таким образом, каждый блок в нашем случае содержит 9*4=36 элементов вектора. Блоки пересекаются, область пересечения составляет 3 ячейки. Всего в изображении умещается 9 блоков. Размерность вектора признаков составляет 324 элемента.

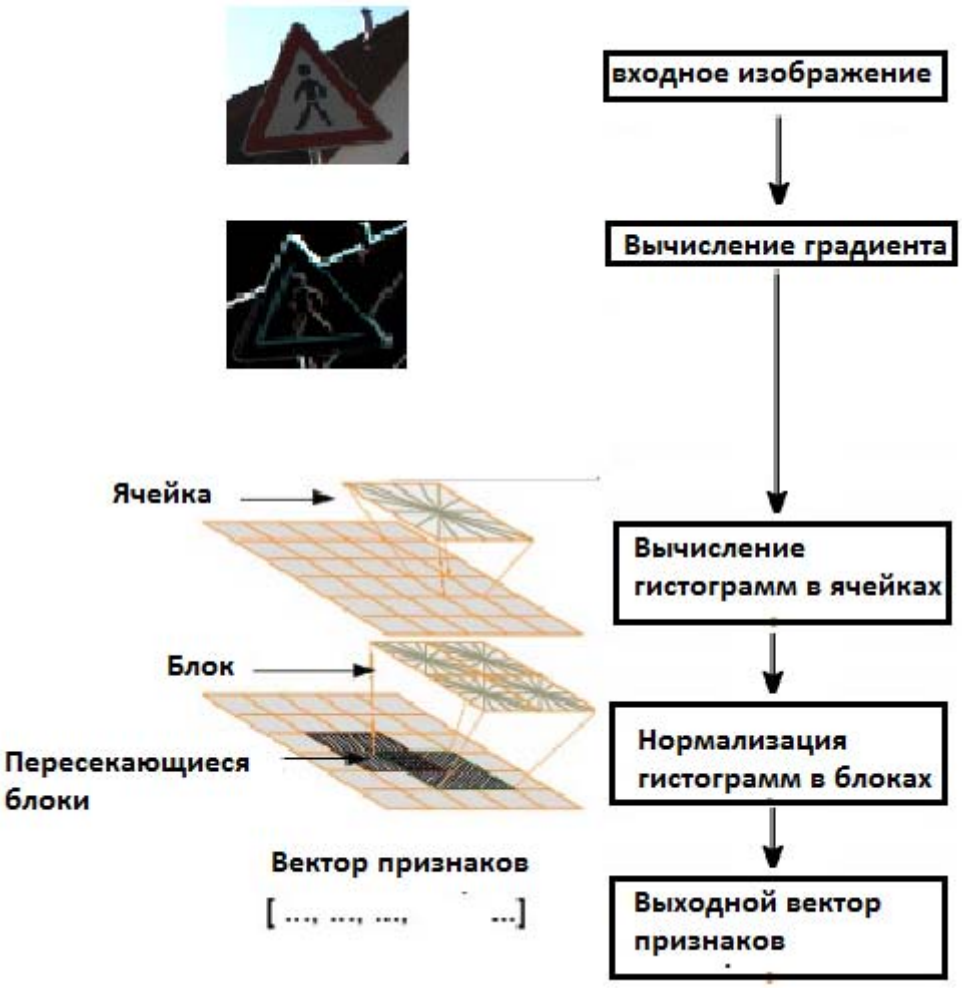


Рисунок 3 – Процесс реализации дескриптора

В процессе было исследовано влияние добавления цветовых компонент в выходной вектор признаков. При этом аналогичным образом формировались цветовые гистограммы, учитывающие присутствие основных цветов в ячейке: красного, синего, белого, черного, желтого. Помимо базовой версии HOG дескриптора существуют различные модификации, алгоритма, применение которых планируется в дальнейшем. Как показывает практика, размерность вектора признаков для высокого процента распознавания может варьироваться от 2000 до 5000. В нашем случае размерность была всего лишь 324.

2 Метод опорных векторов (SVM)

2.1 Описание

SVM - набор методов машинного обучения, используемых для решения задач классификации и регрессии. [4] Стандартный алгоритм SVM принимает входные данные и делает предсказание относительно того к какому из двух классов принадлежат данные. Это соответствует простейшему случаю бинарного линейного классификатора. Предсказание делается на основе модели, которая строится по тренировочной выборке. Тренировочная выборка – это набор векторов признаков и набор меток, которые соответствуют данным векторам. Каждый вектор представляет объект (изображение). Интуитивно модель SVM для векторов размерности 2 можно представить, как плоскость на которой расположены точки, причем точки соответствующие разным классам разделены зазором. (рис. 4). На рисунке представлено двумерное пространство, в котором отображены точками векторы признаков.

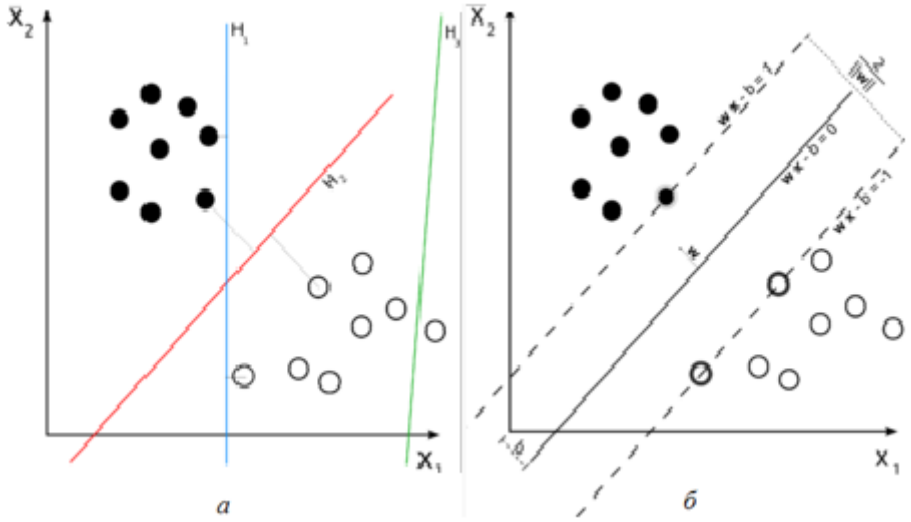


Рисунок 4 – Пространство модели SVM

При классификации предсказание делается исходя из того по какую сторону от разделяющей линии попадет вектор классифицируемого изображения. Оптимизационная задача может быть сформулирована в простых терминах следующим образом: построить такую прямую wx+b=0 которая бы разделяла классы и увеличивала бы геометрический отступ (geometric margin). Здесь под геометрическим отступом понимается расстояние от прямой wx+b=0 до ближайшего элемента из тренировочной выборки. Из (рис. 4, а) видно, что зеленая прямая не разделяет классы, синяя не максимизирует отступ, а красная прямая визуально кажется удовлетворяющей заявленным требованиям. Параметры прямой w и b инвариантны к масштабированию, следовательно, можно построить две параллельные прямые wx+b=1 и wx+b=-1, которые будут проходить через ближайшие (с одной и другой стороны) относительно разделяющей прямой точки (рис. 4, б). Понятие геометрического отступа лежит в основе процесса оптимизации классификатора.

2.2 Оптимизация

Считаем, что задана тренировочная выборка $D = \{(x_i, y_i) | x_i \in R^p, y_i \in \{-1,1\}\}_{i=1}^{n}$ (2)

Для построения оптимального классификатора необходимо решить следующую задачу:

$$\begin{cases} \min \|w\|^2 \\ y^{(i)}(w^T x^{(i)} + b) \geq 1 \end{cases} \quad (3)$$

Оптимизационная проблема может быть представлена как задача нахождения условного экстремума методом множителей Лагранжа:

$$\min_{w,b} \max_{\alpha} \{0.5\|w\|^2 - \sum_{i=1}^{n} \alpha_i [y_i (w \bullet x_i - b) - 1]\} \quad (4)$$

Здесь $\|w\|^2$ - функция, которую мы хотим минимизировать

α_i - множители Лагранжа

$y^{(i)}(w^T x^{(i)} + b) \geq 1 \quad i = \overline{1,n}$ Наложенные ограничения для каждой точки – расстояние от любой точки до плоскости больше либо равно чем геометрический отступ .

Решая задачу квадратичного программирования, мы получаем оптимальные значения w и b :

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \quad (5)$$

Параметр b находится из уравнения плоскости (прямой).

Формула (6) позволяет записать проблему оптимизации в общем виде, учитывая ядро классификатора. В линейном случае оно является просто скалярным произведением.

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - 0.5 \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \quad (6)$$

Решение тут опять же будет иметь вид (5).

Для визуализации и простоты было представлено решение задачи для бинарного классификатора. В общем случае задача решается для n классов, имеющих векторы признаков размерности m. В этом случае в m мерном пространстве строится m-1 мерные гиперплоскости, разделяющие классы.

2.3 Ядра SVM

Случай линейной разделимости встречается далеко не всегда. Для линейно неразделимого набора возможен переход в пространство большей размерности, тогда вероятно, что оптимальная кривая, разделяющая данные в нашем пространстве признаков, будет найдена как гиперплоскость в пространстве более высокой размерности. Для проведения перехода в пространства более высокой размерности используются ядра. Некоторые виды ядер:

- *Polynomial (homogeneous):* $k(x_i, x_j) = (x_i \bullet x_j)^d$
- *Polynomial (inhomogeneous):* $k(x_i, x_j) = (x_i \bullet x_j + 1)^d$
- *Gaussian or Radial Basis Function:* $k(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$
- *Intersection cernel (IK):* $k(x_i, x_j) = \sum_{k=1}^{n} \min(x_i(k), x_j(k))$

3 Эксперименты

Экспериментальные данные были получены при тестировании линейного ядра (osu_svm) и intersection kernel (lib_svm). Второе ядро показало лучший результат. Во время экспериментов была выполнена кросс – валидация [6] (и подбор параметра C классификатора как ее часть). Параметр C олицетворяет компромисс между максимизацией зазора и минимизацией ошибки, в случае линейно неразделимого набора [7]. Подбор параметров осуществляется эмпирическим путем. Была выявлена зависимость процента классификации от числа изображений, содержащихся в тренировочном наборе. Максимальные процент, полученный на немецкой базе, составил 93.6431% (11770/12569) (classification), линейное ядро дало результат 91.1414%. Данные представлены в (таб. 1).

Таблица 1 – Результаты классификации

линейный SVM	**91.14**
IK-SVM	**93.64**

График для изменения результирующего процента от C (рис. 6) для линейного ядра.

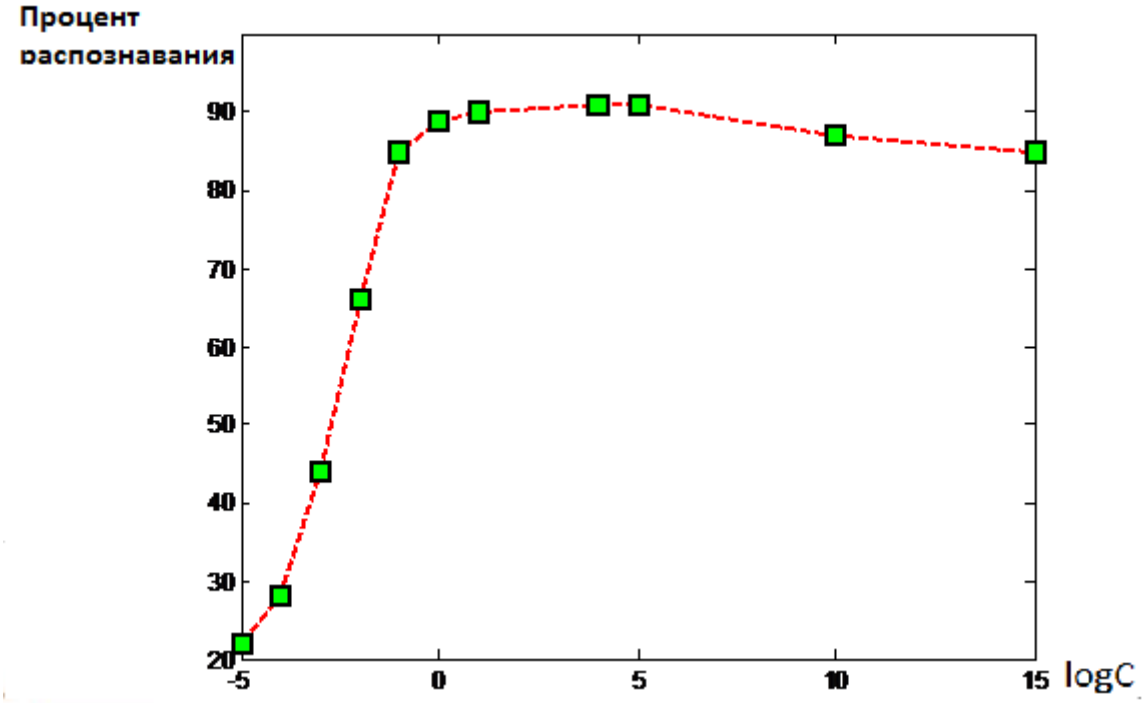


Рисунок 5 – Параметр C для линейного ядра

Под результирующим процентом понимается количество верно классифицированных изображений, отнесенное к общему числу изображений в тренировочной выборке и умноженное на 100%.

4 Практическая реализация

Для практической реализации данный алгоритм должен быть доработан, требуемый процент распознавания стремится к 100%. В случае успеха, алгоритм может быть использован в системах обеспечения безопасности на дороге. Данные системы основаны на детекции и распознавании знаков, пешеходов. При высоком проценте распознавания, система может использоваться для автоматизации движения транспортного средства.

Рисунок 6 – Детектирование дорожных знаков

Так же данный алгоритм может быть использован для автоматизации учета дорожных знаков, который необходим для составления навигационных карт.

Литература

1. Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection, June. 2005. http://ljk.imag.fr/membres/Bill.Triggs/pubs/Dalal-cvpr05.pdf
2. S. Boughorbel, J.-P. Tarel, and N. Boujemaa. Generalized histogram intersection kernel for image recognition. In ICIP, Genova, Italy, 2005. http://perso.lcpc.fr/tarel.jean-philippe/publis/jpt-icip05.pdf
3 . COLOR EXPLOITATION IN HOG-BASED TRAFFIC SIGN DETECTION I.M. Creusen, R.G.J. Wijnhoven, E. Herbschleb, P.H.N. de With http://vca.ele.tue.nl/publications/data/ICreusen2010a.pdf
4. Corinna Cortes and V. Vapnik, "Support-Vector Networks", Machine Learning, 20, 1995. http://www.springerlink.com/content/k238jx04hm87j80g/
5. P. H. Chen, C. J. Lin, and B. Schölkopf, A tutorial on ν-support vector machines, Appl. Stoch. Models. Bus. Ind. 2005, 21, 111-136.
6. http://genome.tugraz.at/proclassify/help/pages/XV.html
7. http://www.svms.org/parameters/
8. http://www.csie.ntu.edu.tw/~cjlin/libsvm/