Высокопроизводительное моделирование распространения электромагнитного поля на графических процессорах с гибким использованием ресурсов

Жердев Д. А., Фурсов В. А.

Самарский государственный аэрокосмический университет им. академика С. П. Королева,
Институт систем обработки изображений РАН.

Параллельные вычислительные технологии 2014,
Ростов-на-Дону.

Постановка задач и цель работы

- Цель работы: разработка методик и инструментальных средств математического моделирования радиолокационных характеристик техногенных объектов.
- При решении задач распознавания радиолокационных характеристик техногенных объектов на подстилающей поверхности важным требованием к программе моделирования эталонов является оперативность получения результата, а так же его точность.
- При обработке данных на графическом процессоре возникает задача обработать как можно больше данных за меньшее количество пересылок данных из оперативной памяти компьютера.

- 1. Процесс моделирования.
- 2. Два вида преобразования ближнего поля в дальнее.
- 3. Особенности реализации алгоритма с гибким использованием ресурсов на CUDA.
- 4. Заключение.

Процесс моделирования

- В данной работе рассматривается задача определения поля рассеяния объекта. В изотропной и недиспергирующей, а так же линейной и свободной от сторонних электрических зарядов среде распространение электромагнитного поля можно описывать двумя уравнениями Максвелла

$$rot\vec{E} = -\mu_0\mu\frac{\partial\vec{H}}{\partial t},$$

$$rot\vec{H} = \sigma\vec{E} + \varepsilon_0\varepsilon\frac{\partial\vec{E}}{\partial t},$$

- с учетом граничных условий

$$E_\tau^{(1)} - E_\tau^{(2)} = 0,$$

$$H_n^{(1)} - H_n^{(2)} = 0.$$

Итерационная схема вычисления ***x*** компоненты вектора напряженности электромагнитного поля

- В данной работе исследуется метод конечных разностей решения уравнений Максвелла. Конечный набор точек, в которых будут рассчитаны приближенные значения, соответствует дискретной сетке
- $(x, y, z, t) = (i\Delta x,\ j\Delta y, k\Delta z, n\Delta t)$,
- где - шаги по пространству вдоль оси x, y, z и по времени t соответственно.

$$H_x^{n+\frac{1}{2}}(i, j+\frac{1}{2}, k+\frac{1}{2}) = \frac{\Delta t}{\mu\mu_0}\left(\frac{E_y^n(i, j+\frac{1}{2}, k+1) - E_y^n(i, j+\frac{1}{2}, k)}{\Delta z} - \frac{E_z^n(i, j+1, k+\frac{1}{2})}{\Delta y} + \right.$$

$$\left. + \frac{E_z^n(i, j, k+\frac{1}{2})}{\Delta y}\right) + H_x^{n-\frac{1}{2}}(i, j+\frac{1}{2}, k+\frac{1}{2}),$$

$$E_x^n(i+\frac{1}{2}, j, k) = \frac{\Delta t}{\varepsilon\varepsilon_0}\left(\frac{H_z^{n-\frac{1}{2}}(i+\frac{1}{2}, j+\frac{1}{2}, k) - H_z^{n-\frac{1}{2}}(i+\frac{1}{2}, j-\frac{1}{2}, k)}{\Delta y} - \right.$$

$$\left. - \frac{H_y^{n-\frac{1}{2}}(i+\frac{1}{2}, j, k+\frac{1}{2})}{\Delta z} + \frac{H_y^{n-\frac{1}{2}}(i+\frac{1}{2}, j, k-\frac{1}{2})}{\Delta z} - \sigma E_x^{n-1}(i+\frac{1}{2}, j, k)\right) + E_x^{n-1}(i+\frac{1}{2}, j, k).$$

Преобразование ближнего поля в дальнее с расточительным использованием ресурса памяти

- На фиктивной границе, представленной на рис. 1, получаем распределение поля.

$$E_{\tau}^{(1)} - E_{\tau}^{(2)} = -M,$$

$$H_{n}^{(1)} - H_{n}^{(2)} = J.$$

- Запишем уравнения Максвелла в виде:

$$rot\mathbf{E} = -\mathbf{M} - \mu_0 \frac{\partial \mathbf{H}}{\partial t},$$

$$rot\mathbf{H} = \mathbf{J} + \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t}.$$

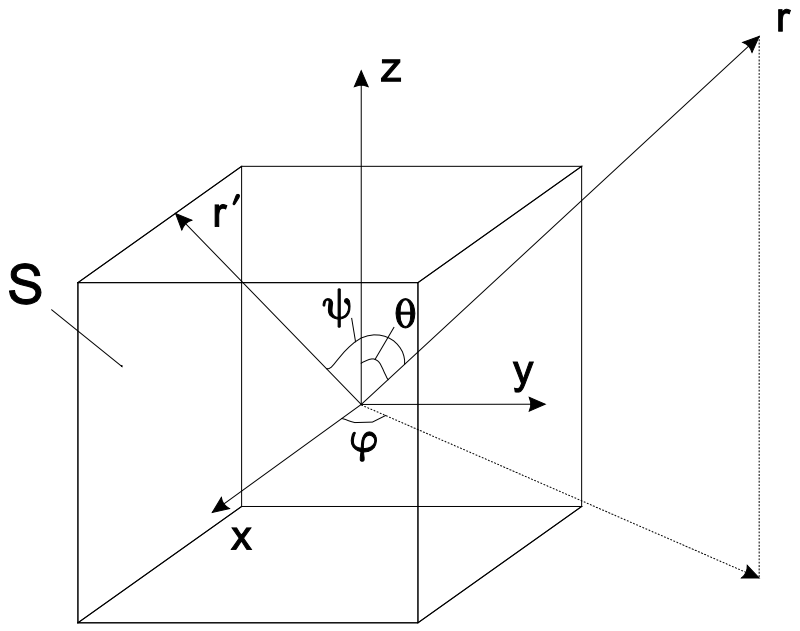


Рис. 1 – Положение дальней точки в сферической системе координат

Выражения для нахождения поля в дальней точке

- Фурье компоненты электромагнитного поля определенные на фиктивной границе $\hat{\mathbf{E}}(\mathbf{r},\omega),\hat{\mathbf{H}}(\mathbf{r},\omega),\hat{\mathbf{M}}(\mathbf{r},\omega),\hat{\mathbf{J}}(\mathbf{r},\omega)$.
- В данных переменных поле в дальней точке запишется как:

$$\mathbf{H}=\frac{ike^{-ikr}}{4\pi r}\begin{pmatrix}0\\-F_\varphi(\widehat{\mathbf{M}})-Z_0F_\theta(\hat{\mathbf{J}})\\F_\theta(\widehat{\mathbf{M}})-Z_0F_\varphi(\hat{\mathbf{J}})\end{pmatrix},$$

$$\mathbf{E}=\frac{ike^{-ikr}}{4\pi r}\begin{pmatrix}0\\F_\varphi(\hat{\mathbf{J}})-Z_0^{-1}F_\theta(\widehat{\mathbf{M}})\\-F_\theta(\hat{\mathbf{J}})-Z_0^{-1}F_\varphi(\widehat{\mathbf{M}})\end{pmatrix},$$

- где функции $F_\varphi(\vec{a})$ и $F_\theta(\vec{a})$:

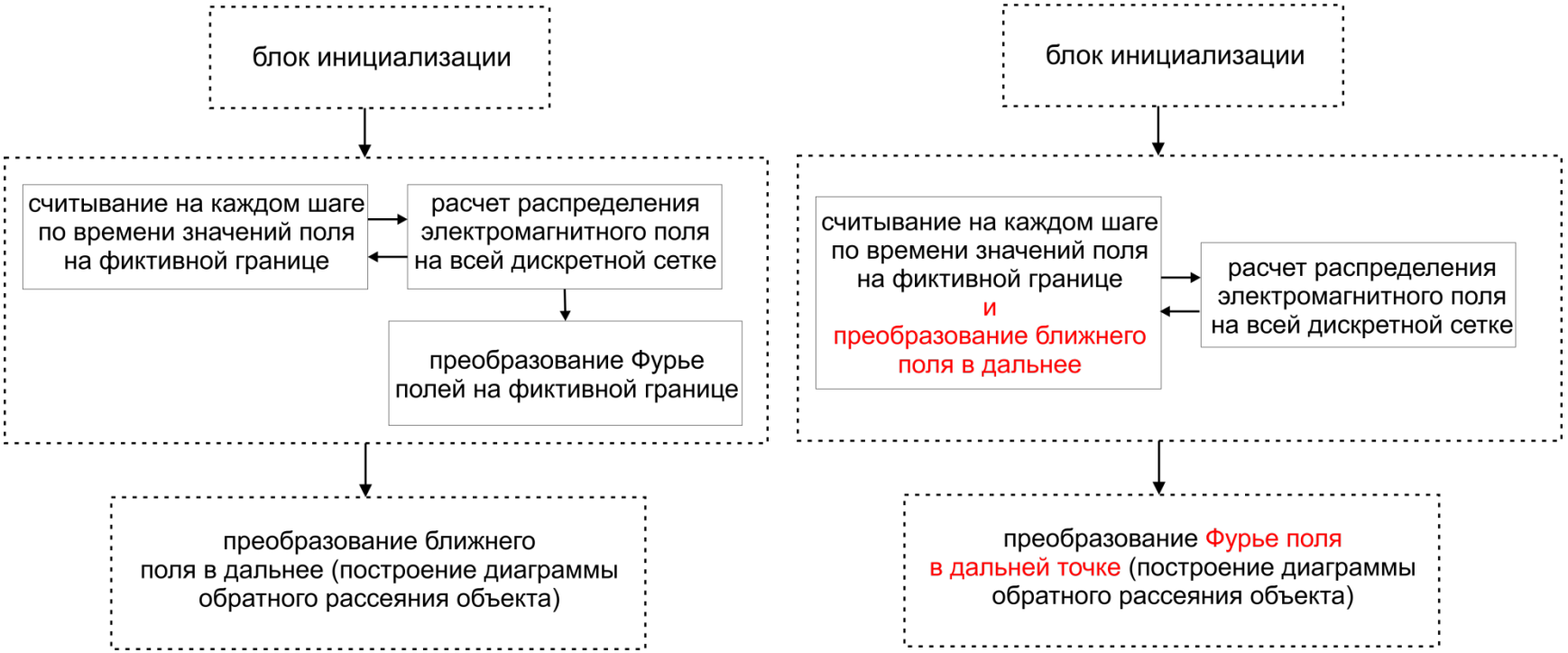
$$F_\varphi(\mathbf{a})=\iint(-a_x\sin\varphi+a_y\cos\varphi)e^{ikr'\cos\psi}dS,$$

$$F_\theta(\mathbf{a})=\iint(a_x\cos\theta\cos\varphi+a_y\cos\theta\sin\varphi-a_z\sin\theta)e^{ikr'\cos\psi}dS,$$

- и $Z_0=\sqrt{\mu_0/\varepsilon_0}$ - импеданс свободного пространства

Две схемы реализации алгоритма

Две схемы реализации алгоритма



Выражения для расчета поля в дальней точке с гибким использованием ресурсов

- Теперь формулы получения поля в дальней точке будут иметь вид:

$$\mathbf{H} = \frac{1}{4\pi rc}\begin{pmatrix} 0 \\ -G_{\varphi}(\mathbf{M}(\tau)) - Z_0 G_{\theta}(\mathbf{J}(\tau)) \\ G_{\theta}(\mathbf{M}(\tau)) - Z_0 G_{\varphi}(\mathbf{J}(\tau)) \end{pmatrix},$$

$$\mathbf{E} = \frac{1}{4\pi rc}\begin{pmatrix} 0 \\ G_{\varphi}(\mathbf{J}(\tau)) - Z_0^{-1} G_{\theta}(\mathbf{M}(\tau)) \\ -G_{\theta}(\mathbf{J}(\tau)) - Z_0^{-1} G_{\varphi}(\mathbf{M}(\tau)) \end{pmatrix},$$

- а функции $G_{\theta}(\vec{a})$ и $G_{\varphi}(\vec{a})$:

$$G_{\varphi}(\mathbf{a}) = \frac{\partial}{\partial t}\oiint(-a_x(\tau)\sin\varphi + a_y(\tau)\cos\varphi)dS,$$

$$G_{\theta}(\mathbf{a}) = \frac{\partial}{\partial t}\oiint(a_x(\tau)\cos\theta\cos\varphi + a_y(\tau)\cos\theta\sin\varphi - a_z(\tau)\sin\theta)dS.$$

- Здесь аргумент :

$$\tau = t - \frac{r - r'\cos\psi}{c}$$

- определяет время задержки отклика поля на фиктивной границе в дальнюю точку

Разностные выражения для нахождения поля в дальней точке с гибким использованием ресурсов

- Определение дальнего поля:

$$\mathbf{E}^{n+1/2+f} = \frac{1}{4\pi rc}\begin{pmatrix} 0 \\ G_\varphi(\mathbf{J}^n) - Z_0^{-1} G_\theta(\mathbf{M}^{n+1/2}) \\ -G_\theta(\mathbf{J}^n) - Z_0^{-1} G_\varphi(\mathbf{M}^{n+1/2}) \end{pmatrix}$$

$$\mathbf{H}^{n+f} = \frac{1}{4\pi rc}\begin{pmatrix} 0 \\ -G_\varphi(\mathbf{M}^{n+1/2}) - Z_0 G_\theta(\mathbf{J}^n) \\ G_\theta(\mathbf{M}^{n+1/2}) - Z_0 G_\varphi(\mathbf{J}^n) \end{pmatrix}$$

- Время задержки отклика поля на фиктивной границе:

$$f_n = \frac{r - r'\cos\psi}{c\Delta t}$$

- Функции входящие в определение дальнего поля:

$$G_\varphi(\mathbf{a}) = \sum_j \sum_k \left(-\frac{a_x^{n+1}(x_0, j, k) - a_x^n(x_0, j, k)}{\Delta t}\sin\varphi + \frac{a_y^{n+1}(x_0, j, k) - a_y^n(x_0, j, k)}{\Delta t}\cos\varphi\right)\Delta y \Delta z$$

$$G_\theta(\mathbf{a}) = \sum_j \sum_k \left(\frac{a_x^{n+1}(x_0, j, k) - a_x^n(x_0, j, k)}{\Delta t}\cos\theta\cos\varphi + \frac{a_y^{n+1}(x_0, j, k) - a_y^n(x_0, j, k)}{\Delta t}\cos\theta\sin\varphi - \frac{a_z^{n+1}(x_0, j, k) - a_z^n(x_0, j, k)}{\Delta t}\sin\theta\right)\Delta y \Delta z.$$

Особенности реализации алгоритма на CUDA

- Создание массивов f_X, f_Y, f_Z размерами cubeY*cubeZ, cubeY*cubeZ, cubeX*cubeY соответственно будет храниться величина задержки τ_n, где cubeX, cubeY, cubeZ – линейные размеры фиктивной границы.

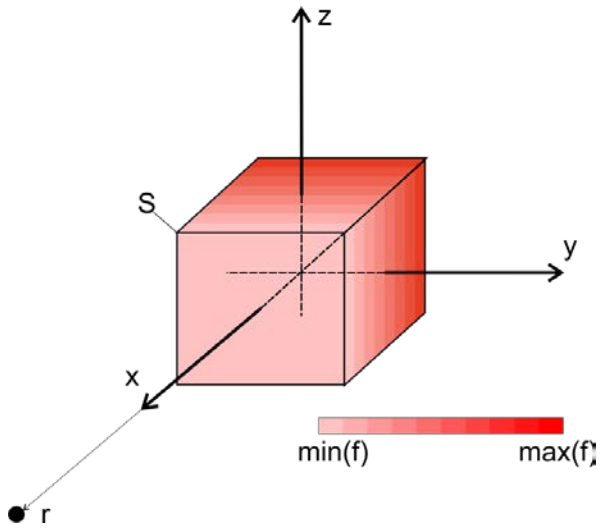


Рис. 2 – Диаграмма распределения величины f по фиктивной границе

Листинг вычисления поля в дальней точке с гибким использованием ресурсов (этап 1)

```
unsigned int bx = blockIdx.x;
extern __shared__ float Jy[];
extern __shared__ float Jz[];
extern __shared__ float My[];
extern __shared__ float Mz[];
extern __shared__ int delay[];

Jy[threadIdx.x] = JyX[k + bx*2 + threadIdx.x*2*cubeY];
Jz[threadIdx.x + cubeZ] = JzX[k + bx*2 + threadIdx.x*2*cubeY];
My[threadIdx.x + 2*cubeZ] = MyX[k + bx*2 + threadIdx.x*2*cubeY];
Mz[threadIdx.x + 3*cubeZ] = MzX[k + bx*2 + threadIdx.x*2*cubeY];
delay[threadIdx.x + 4*cubeZ] = f_X[bx + threadIdx.x*cubeY + k*cubeY*cubeZ + n*2*cubeY*cubeZ];
__syncthreads();

if(threadIdx.x == 0){
        for(unsigned int j = 0; j < blockDim.x; j++){
         Hteta [bx + delay[j + 4*cubeZ]*blockDim.x] += g(0, Jy[j], 0, My[j + 2*cubeZ], Mz[j + 3*cubeZ], fi, teta);
         Hfi [bx + delay[j + 4*cubeZ]*blockDim.x] += g(0, Jy[j], Jz[j + cubeZ], 0, My[j + 2*cubeZ], fi, teta);
        }
}
```

Листинг вычисления поля в дальней точке с гибким использованием ресурсов (этап 2)

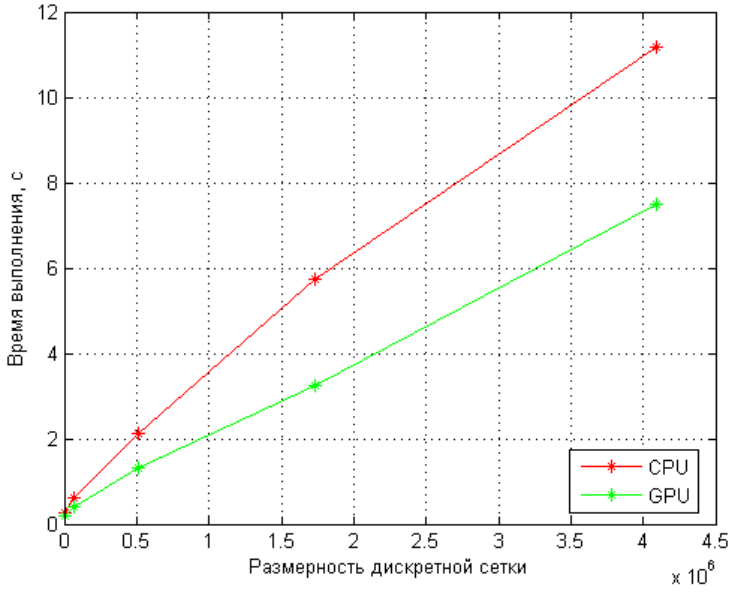
```
unsigned int bx = blockIdx.x;
extern __shared__ float sHteta[];
extern __shared__ float sHfi[];

sHteta[threadIdx.x] = HtetaTemporary[threadIdx.x + bx*cubeY];
sHfi[threadIdx.x + cubeY] = HfiTemporary[threadIdx.x + bx*cubeY];
__syncthreads();

if(threadIdx.x == 0){
        for(unsigned int j = 0; j < blockDim.x; j++){
          Hteta[timeCount + bx + n*timeStep] = cuCaddf(Hteta[timeCount + bx + n*timeStep], (sHteta[j]));
          Hfi[timeCount + bx + n*timeStep] = cuCaddf(Hfi[timeCount + bx + n*timeStep], (sHfi[j + cubeY]));
          HtetaTemporary[j + bx*cubeY] = 0;
          HfiTemporary[j + bx*cubeY] = 0;
        }
}
```

Оценка быстродействия

- Для оценки быстродействия программного комплекса были проведены сравнительные эксперименты с использованием CPU и GPU. Численные эксперименты на GPU проводились на видеокарте Nvidia GeForce 8800 GT 1024 Mb RAM, Compute capability 1.1, а так же на CPU AMD Athlon 64 X2 5600+.



Заключение

- Полученные результаты показывают возможность существенного увеличения области моделирования. Это позволит расширить классы моделируемых объектов для решения задач распознавания по радиолокационным портретам.

Спасибо за внимание!